

C. Matching algorithm

Before drilling into the details of the algorithm, one issue needs to be addressed, namely, the quantification of the importance levels. Since the personal valuation to the qualitative values high, medium, and low may vary from a user to another, it may be a good idea to allow the user to set a quantitative value for these levels by themselves. Additionally, the system may offer default values. Yet, it may be argued that since the user will set the quantitative values of the importance levels by themselves, why not use quantitative values in the rule itself. We believe that humans are more accustomed to working with qualitative values even though the perception of the magnitude of these qualitative values may differ from someone to the other.

As such, the score, a provider's policy receives is:

$$S_j = W_1 X_j + W_2 Y_j + W_3 Z_j$$

Where:-

S_j is the total score received by the policy of provider j .

W_1 , W_2 , and W_3 are the quantitative values assigned by the user corresponding to the importance levels high, medium, and low.

X_j , Y_j , and Z_j are the numbers of satisfied rules from the high, medium, and low importance category in order.

So as for scoring and selection algorithm to function properly, the following assumptions are set ahead: (1) the process containing the algorithm will be triggered as soon as the handset is searching for a roaming network; (2) the algorithm will terminate after a specified period t and; and (3) a dirty flag will change its value to *True* upon arrival of a new provider's policy.

The algorithm includes a main function that initializes the return values *BP* (best policy identifier), *Score [BP]* (the score of the best policy), and *NS [BP]* the list of rules with high importance level that were not satisfied as well as the policy identifier j . The function then counts down the period t and checks for the dirty flag until it terminates. If a provider's policy is found, two functions are called; the first function scores the policy and the second one computes whether it is the best policy.

The scoring function *ScoreP* is assumed to match a user's rule to a provider's rule by parsing both of them from their XML files. The matching is conducted by plugging the value v_i from the provider's rule, match it according to the mathematical operator, and score it. The function continues by accumulating the scores of the matched rules. Additionally, if a high importance rule is not satisfied, its identifier is added to the list *NS [j]*.

The selection function *SelectBP* compares the score of the policy that has just completed scoring and set it as the best policy if it scores higher than the current best policy. If the scores are equal, the best policy is selected based on the number of unsatisfied high importance rules. A provider's policy that has lower number of unsatisfied high importance rules is favored.

ALGORITHM 1 Policy Scoring and Selection

INPUT: A user rule set *PU*, a provider rule set *PP*, a time interval t , and three weights W_1 , W_2 , and W_3

OUTPUT: A best policy *BP*, its score *Score[BP]*, and a list none satisfied important rules *NS*.

FUNCTION main

```
    terminateTime  $\leftarrow$  currentTime() + t
    j  $\leftarrow$  0
    BP  $\leftarrow$  0
    Score [BP]  $\leftarrow$  0
    NS [BP]  $\leftarrow$  0
    WHILE currentTime() < terminateTime DO
        IF dirtyFlag THEN
            PP [j]  $\leftarrow$  PP
            j  $\leftarrow$  j+1
            Call ScoreP (PP [j], W1, W2, W3)
            Call SelectBP (Score [BP], NS [BP], Score [j])
            Return BP, Score [BP], NS [BP]
```

```

ENDIF
FUNCTION ScoreP
   $Score[j] \leftarrow 0$ 
  WHILE  $rulesLeft > 0$  DO
    Extract a new rule  $R_{U_i}$  from  $P_U$ 
    Match  $R_{U_i}$  to  $R_{P_i}$ 
    IF rule is matched THEN
      Add score to  $Score[j]$ 
    ELSEIF importance = "High"
      Add rule identifier to  $NS[j]$ 
    ENDIF
  Return Score [ $j$ ], NS [ $j$ ]
FUNCTION SelectBP
  IF  $Score[j] > Score[BP]$  THEN
     $BP \leftarrow j$ 
     $Score[BP] \leftarrow Score[j]$ 
     $NS[BP] \leftarrow NS[j]$ 
  ELSEIF  $Score[j] = Score[BP]$  AND  $NS[j] < NS[BP]$  THEN
     $BP \leftarrow j$ 
     $Score[BP] \leftarrow Score[j]$ 
     $NS[BP] \leftarrow NS[j]$ 
  ENDIF
  Return BP, Score [ $BP$ ], NS [ $BP$ ]

```
