

distribute the subrelations to the extra sites. The algorithm determines for each relation R_i the partitioning level, denoted by l_i , that should be used. The partitioning level indicates the number of fragments into which R_i should be divided. Each iteration of the algorithm considers the addition of one extra site if the savings obtained in local processing costs outweigh the additional data partitioning costs. The complete partitioning algorithm is given below.

Partitioning Algorithm(PART):

```

for  $i = 1$  to  $r$  do
   $l_i = 1$ ; /*  $l_i$  is the partitioning level of  $R_i$  */
  for  $k = 1$  to  $(n - r)$  do begin
    for  $i = 2$  to  $r$  do
      compute  $\text{Gain}(R_i^{l_i})$ ;
       $\text{Max\_Gain} = \max_i(\text{Gain}(R_i^{l_i}))$ ;
      if ( $\text{Max\_Gain} > 0$ ) then
         $l_j = l_j + 1$ ; /*  $R_j$  is the relation for which  $\text{Max\_Gain}$  is achieved */
    end;
  for  $i = 1$  to  $r$  do
    partition  $R_i$  into  $l_i$  subrelations of equal size and distribute them to the extra sites;
  where  $\text{Gain}(R_i^{l_i}) = (S_{i,f}^{l_i} + S_{i,b}^{l_i}) - (S_{i,f}^{l_i+1} + S_{i,b}^{l_i+1}) + (PT_i^{l_i} - PT_i^{l_i+1})$ .
          $S_{i,f}^{l_i}$  and  $S_{i,b}^{l_i}$  are the local processing costs at site  $S_i$  in forward and backward
         reduction given that  $R_i$  is at partitioning level  $l_i$ ;  $PT_i^{l_i}$  stands for the partitioning
         cost.

```

Given n sites and r relations (with $n > r$), the CPU time complexity of the algorithm is $O((n - r) * r)$.

Example 6. Let us consider the query $R_1 \bowtie R_2 \bowtie R_3 \bowtie R_4$, and let us assume that there are 6 sites in the network. Initially all relations are at partition level 1. The gains to be computed are $(S_{i,f}^1 + S_{i,b}^1) - (S_{i,f}^2 + S_{i,b}^2) - PT_i^2$ for $i = 2, 3, 4$. Note that PT_i^2 denotes the cost to obtain a partition containing half the tuples of R_i at a new site. Let us assume that Max-gain is the gain obtained by partitioning R_3 into two equal sized partitions, one at its original site S_3 and the second at the extra site S_5 . Since we have one additional site available, the actual partitioning does not occur yet, but we update the partitioning level of R_3 to 2. In the next iteration the gains obtained by partitioning R_2 and R_4 are unchanged, but $\text{Gain}(R_3^2)$ becomes: $(S_{3,f}^2 + S_{3,b}^2) - (S_{3,f}^3 + S_{3,b}^3) + (PT_3^2 - PT_3^3)$. Now PT_3^3 denotes the cost of obtaining a partition of R_3 containing one third of the tuples of R_3 . It is important to observe here that we should only consider subdividing a relation into fragments of equal size. A partitioning into unequal fragments will imply that the largest fragment becomes the bottleneck with regard to local processing costs. Let us assume that $\text{Gain}(R_3^2)$ is largest among the gains considered. Thus the algorithm terminates with R_3 at partitioning level 3 and all other relations at partitioning level 1. This means that the additional sites S_5 and S_6 will each receive a fragment of R_3 whose cardinality is $1/3$ of the original cardinality of R_3 .

