Comm_Interval — Time between communication steps
Termination — Tags termination messages
Token — Tags token message
Unexamined_Subproblem — Tags message containing unexamined subproblem

Functions:
Current_Time() — Wall clock time
Delete_Min() — Delete subproblem with least lower bound from priority queue
First_Element() — Returns first element from priority queue without deleting it
Initialize() — Set priority queue size to 0
Insert() — Insert subproblem into priority queue
Is_Empty() — Returns true if priority queue is empty
Lower_Bound() — Returns lower bound associated with unexplored subproblem

Variables:
color — Process color (for termination detection)
global_c — Cost of globally best solution found so far
id — Process rank
initial — Initial problem
last_comm — Time of last communication
local_c — Cost of best solution found so far by this process
local_s — Best solution found so far by this process
msg_count — Messages sent minus messages received
q — Priority queue
token — Token passed around ring for termination detection
u — State space tree node
v — New node with additional constraint

Parallel Best-First Branch and Bound (minimization):

```
Initialize ($q$)
if $id = 0$ then
  Insert ($q$, $initial$)
  $token.c \leftarrow \infty$
  $token.color \leftarrow$ WHITE
  $token.count \leftarrow 0$
  Send $token$ to successor process
endif
$local\_c \leftarrow \infty$
$best\_soln \leftarrow \infty$
$last\_comm \leftarrow$ Current_Time()
$msg\_count \leftarrow 0$
$color \leftarrow$ WHITE
repeat
  if Is_Empty($q$) or (Current_Time()$-last\_comm > Comm\_Interval$) then
    BandB_Communication()
    $last\_comm \leftarrow$ Current_Time()
  else if not Is_Empty($q$) then
    $u \leftarrow$ Delete_Min($q$)
    if Lower_Bound($u$) $< best\_c$ then
      $color \leftarrow$ BLACK
      if $u$ is a solution then
        if Lower_Bound($u$) $< global\_c$ then
          $local\_s \leftarrow u$
          $local\_c \leftarrow$ Lower_Bound($local\_s$)
        endif
```

```
          else
            for $i \leftarrow 1$ to Possible_Constraints($u$) do
              Add constraint $i$ to $u$, creating $v$
              if Lower_Bound($v$) $<$ $global\_c$ then
                Insert($q$, $v$)
              endif
            endfor
          endif
        endif
      endif
    forever

BandBCommunication():
    if there is a pending message with a Termination tag then Halt endif
    if there is a pending message with a Token tag then
      Receive message containing token
      if $local\_c < token.c$ then
        $token.c \leftarrow local\_c$
        $token.s \leftarrow local\_s$
      endif
      if $token.c \leq$ Lower_Bound(First_Element($q$)) then Initialize($q$) endif
      $global\_c \leftarrow token.c$
      if $id = 0$ then
        if ($color$ = WHITE) and ($token.color$ = WHITE) and
            ($token.count + msg\_count = 0$) then
          Send messages with a Termination tag to all other processes
          Halt
        else
          $token.color \leftarrow$ WHITE
          $token.count \leftarrow 0$
        endif
      else
        if $color$ = BLACK then $token.color \leftarrow$ BLACK
        $token.count \leftarrow token.count + msg\_count$
      endif
      Send token to successor
      $color \leftarrow$ WHITE
    endif
    while there are pending messages with tag Unexamined_Subproblem do
      Receive message with unexamined subproblem $u$
      $msg\_count \leftarrow msg\_count - 1$
      $color \leftarrow$ BLACK
      if Lower_Bound($u$) $<$ $global\_c$ then Insert ($q$, $u$)
    endwhile
    if there is more than one unexamined subproblem in $q$ then
      Send unexamined subproblem to another process
      $msg\_count \leftarrow msg\_count + 1$
      $color \leftarrow$ BLACK
    endif
    return
```