```
! A one machine sequencing problem,
also known as the walking + biking problem.
Given one bicycle, and N persons,
and a distance they all must cover,
starting at the same time,
how much time should each person spend walking
and biking so they all cover the distance in minimum time;
!Ref: Chvatal, V.(1983), "On the Bicycle Problem,"
Discrete Applied Mathematics, North-Holland Publishing Company, 5 165-173
! Keywords: Bicycle, Biking, Chvatal, LINGO, One machine, Scheduling, Sequencing, Walking;
SETS:
  PERSON: W, B, X, U, Y, Z, Sorder;
ENDSETS
DATA:
! Case 1. From Chvatal. The lower bound (of 55)
  is tight for this data set.
! Input parameters;
!  Distance to be covered;
!Case01;  D = 100;
!  Walking speeds;
!Case01;  W = 1 2 1;
!  Bicycling speeds;
!Case01;  B = 6 8 6;

! Case 2. From Chvatal. The lower bound (of 10)
  is not tight for this data set. We cannot
  generate a feasible Walk->Bike->Walk schedule
  that achieves the LP bound;
!Input parameters;
!  Distance to be covered;
!Case02  D = 90;
!  Walking speeds;
!Case02  W = 13  13   3   3;
!  Bicycling speeds;
!Case02  B = 27  27  18  18;

!Case 3;
! Input parameters;
!  Distance to be covered;
!Case03  D = 100;
!  Walking speeds;
!Case03  W = 1 2 3;
!  Bicycling speeds;
!Case03  B = 6 8 9;


!Case 4;
! Input parameters;
!  Distance to be covered;
!Case04  D = 100;
!  Walking speeds;
!Case04  W = 1 1 1;
!  Bicycling speeds;
!Case04  B = 3 3 3;

ENDDATA
! Variables for each person i:
    X( i) = total time walking forward,
    U( i) = total time walking backward,
    Y( i) = total time biking forward,
    Z( i) = total time biking backward,
;
SUBMODEL BikeWalk:
! This model contains constraints on an aggregate version
  of the bike & walk problem.  Any complete detailed solution to the problem
  must satisfy at least these aggregate constraints,
  so the solution to this problem provides a lower bound;
MIN = T;
```

```
@FOR( PERSON( i):
 ! Total travel time of person i  <= T;
   X( i) + U( i) + Y( i) + Z( i) <= T;
 ! Net distance of person i = D.  Must get to destination;
   W( i)* X( i) - W( i) * U( i) + B( i) * Y( i) - B( i) * Z( i) = D;
    );

 ! Cannot use the bicycle more that total time T;
   @SUM( PERSON( i): Y( i) + Z( i)) <= T;

 ! In net the bicycle goes no further than D;
   @SUM( PERSON( i): B( i) * Y( i) - B( i) * Z( i)) <= D;
ENDSUBMODEL

CALC:
! Solve the aggregate, relaxed problem;
!  @GEN( BikeWalk); ! Generate the scalar version of model;
  @SOLVE( BikeWalk);
  @WRITE( ' Total time= ', T, @NEWLINE(1));
  @WRITE( '     Distances', @NEWLINE(1));
  @WRITE(' Person   Walk+     Walk-    Bike+     Bike-', @NEWLINE( 1));
  @FOR( PERSON( i):
    @WRITE('   ', i, ' ', @FORMAT( W( i)* X( i),'9.3F'), ' ', @FORMAT( W( i) * U( 
i),'9.3f'),
        ' ',  @FORMAT( B( i) * Y( i),'9.3f'), ' ',  @FORMAT( B( i) * Z( i), '9.3f'), 
@NEWLINE(1));
      );

  @WRITE( @NEWLINE( 1),'     Times', @NEWLINE(1));
  @WRITE(' Person   Walk+     Walk-    Bike+     Bike-', @NEWLINE( 1));
  @FOR( PERSON( i):
    @WRITE('   ', i, ' ', @FORMAT( X( i),'9.3F'), ' ', @FORMAT( U( i),'9.3f'),
        ' ',  @FORMAT( Y( i),'9.3f'), ' ',  @FORMAT( Z( i), '9.3f'), @NEWLINE(1));
      );

! Do postprocessing to (hopefully) get a feasible detailed solution
  that achieves the lower bound, and is thus optimal;
! We restrict ourselves to Walk->Bike->Walk schedules, i.e., each person
 first walks to the bike, then bikes for awhile(perhaps backwards), and
 then walks the remaining distance.
 The detailed schedule is feasible if the person i-1 finishes its use of the bike
 before the person i, who needs the bike, arrives at the bike position;
  @WRITE( @NEWLINE( 1), ' The detailed schedule:', @NEWLINE( 1));
! Choose a sort order. Put fast cyclists first;
  Sorder = @SORT( - B);
! But if fast cyclist goes backwards, do not put first;
  @IFC( Z( sorder( 1)) #GT# 0:
    temp = Sorder( 2);  ! Swap with #2;
    Sorder( 2) = Sorder( 1);
    Sorder( 1) = temp;
      );
  BikeAt = 0;     ! Initial position of Bike;
  BFtimePrv = 0; ! Time available of Bike;
! Loop over the persons, computing their
  Walk, Bike, Walk positions and times;
  @FOR( PERSON( i):
     si = Sorder( i); ! Use sort order;
! Arrive at Bike time after first walk;
     ATime = BikeAt/ W( si);
! Calculate bike ride incremental distance;
     Bdist = B( si) * Y( si) - B( si) * Z( si);
! Calculate bike ride incremental time;
     Btime =  Y( si) + Z( si) ;
! Ending position of bike for person i;
     Bend = BikeAt + Bdist;
! Ending time of bike for person i;
     BFtime = Atime + Btime;
    @WRITE( ' Person ',  PERSON( si),
```

```
                         ' walk to ', @FORMAT( BikeAt,'12.4f'),' Time= ', @FORMAT( Atime,
'12.4f'),
                        '  Bike to ', @FORMAT( Bend,'12.4f'),  ' Time= ', @FORMAT( BFtime,
'12.4f'),
                        '  Walk to ', @FORMAT( D,'12.4f'),  ,  @NEWLINE( 1));
! Check if feasible, i.e., person i-1 finishes bike use before i needs it;
    @IFC( BFtime #LT# BFtimePrv:
      @WRITE( 'Schedule not feasible', @NEWLINE( 1));
        );
   BFtimePrv = BFtime; ! Get ready for next i;
! And we leave the bike for next person at;
    BikeAt = Bend;
      );
ENDCALC
```