El único hombre que nunca ha cometido un error es aquel que nunca ha hecho nada.

Theodore Roosevelt

10. Conclusiones.

I. En lo referente a la Filosofía de la Ciencia de la Administración:

1. La Ciencia de la Administración (OR/MS) es comunicada a través de la modelación mediante relaciones lógicas las cuales proveen justificaciones verificables para las inferencias producidas.
2. A pesar de esto, lo anterior no contribuye a evidenciar el proceso psicológico mediante el cual las inferencias son desarrolladas por primera ocasión lo cual permanece siendo básicamente intuitivo, ya que el proceso para abstraer un problema administrativo en uno científico sigue siendo un arte.
3. Una buena práctica en la modelación matemática en el OR/MS es aplicando un enfoque evolutivo mediante una secuencia de modelos de alcance y complejidad incremental. Deliberadamente se omiten temporalmente consideraciones del problema. Si el modelo se mantiene tratable entonces se continua el proceso de enriquecimiento de otro modo se modifican y/o simplifican los supuestos.
4. La clave en la aplicación del OR/MS es la identificación de aquellas consideraciones que permitan abstraer una realidad y que a la vez mantengan su solución computacional de manera tratable.

II. En lo referente a la aplicación práctica de la Ciencia de la Administración:

1. El aprendizaje del OR/MS implica el ejercicio de un pensamiento activo y crítico mediante el cual se vincula el conocimiento nuevo con el previo mediante una relación no memorizada, sino construida, contribuyendo al desarrollo de abstracciones con un significado tangible que es posible transferir a diversas situaciones para solucionar nuevos problemas.
2. En pocos profesionistas se puede afirmar que el OR/MS ha jugado un rol preponderante en sus carreras profesionales. Menos del 6% de los profesionistas de las empresas del AMM aplican las matemáticas en la toma de decisiones en general.
3. Existe una falta de aplicación del OR/MS en la administración de las empresas en general y en la logística en particular. Solo 2 empresas de las 55 muestreadas (menos del 5%) afirmaron utilizar como estrategia de solución para sus problemas de logística el desarrollo de modelos matemáticos.
4. La extrapolación del pasado es la principal herramienta para la toma de decisiones logística en las empresas del AMM. El 32% de las empresas muestreas del AMM aplican tan solo la experiencia.

III. En lo referente a la oferta de software comercial disponible actualmente para las empresas:

1. La problemática de la logística actualmente está siendo atendida por dos habilitadores: (1) el OR/MS y (2) la oferta de software computacional (sistemas ERP's y APS's).
2. Las investigaciones realizadas indican la falta de rigor matemático de los métodos de optimización encontrados en la oferta comercial de software APS. Solo se incluyen heurísticas poco sofisticadas.
3. El análisis comparativo referente a la oferta de software comercial APS para atender nuestro problema de investigación, evidencia lo improbable que resulta el que una sola alternativa de software "genérico" logre incluir todos los aspectos particulares que requieren ser considerados para los diversos escenarios a tener que ser considerados en la logística.
4. Tan solo 7 de las 55 empresas muestreadas del AMM han intentado resolver el problema de ruteo a través de la implementación de sistemas APS's. No reportan éxito a cabalidad.
5. Mientras las empresas no asocien y a la vez diferencien el uso de los sistemas ERP/APS con la aplicación del OR/MS, habrá un pobre aprovechamiento de las inversiones hechas en los primeros.
6. La aplicación del OR/MS puede mitigar la falta de funcionalidad encontrada en los sistemas APS.

IV. En lo referente al planteamiento del problema y a la estrategia de solución propuesta:

1. De las 55 empresas muestreadas del AMM, casi el 20% enfrentan variantes similares a la tratada como objetivo en nuestro proyecto de investigación relacionada a la logística de ruteo.
2. El problema planteado usualmente en la práctica requiere ser atendido con instancias de gran escala ($\geq$70 clientes) y con un fuerte porcentaje de ventanas de horario activas ($\geq$90%). Además las estructuras de amplitud de las ventanas de horario también es muy alto ($\geq$75%) lo cual evidencia el innegable compromiso real en el tema del servicio al cliente.
3. Nuestro problema de investigación es NP-Hard y apunta hacia una de las más ricas clases de problemas combinatorios en el OR/MS. La variante del problema está diferenciada de investigaciones previas a la vez que no existen extensamente aplicaciones de algoritmos genéticos.
4. La complejidad matemática intrínseca en el planteamiento de nuestro problema de investigación hace prever la necesidad de desarrollar algoritmos de solución que puedan ofrecer soluciones razonablemente buenas en tiempos de ejecución computacional cortos.
5. Aunque nuestro problema parte de una circunstancia orientada a la actividad humana (ciencias sociales), nuestra propuesta de investigación hizo uso de las ciencias exactas, las ciencias naturales y las ciencias computacionales. Los resultados son satisfactorios.

V. En lo referente a los resultados obtenidos en el experimento para la prueba de la hipótesis:

1. El Algoritmo de Ramificación y Corte BC que se aplica como instrumento para el Grupo Control en el diseño experimental, explota eficientemente la estructura matemática del problema logrando alcanzar la solución óptima para las instancias del problema descritas como "sencillas", las cuales tienen características comparables a las que Ascheuer, Fischetti, y Grotschel exponen en su investigación (2001).
2. Además, nuestro Algoritmo BC obtiene soluciones "al menos" muy cercanas al óptimo para las instancias que son particularmente difíciles de resolver y en las cuales el proyecto de investigación se concentra.
3. La hipótesis del proyecto de investigación es comprobada:

   Podemos establecer a un 100% de confiabilidad que el algoritmo genético propuesto alcanza un porcentaje de optimalidad > ***90%*** en un tiempo computacional ***≤5 minutos.***
4. También podemos establecer que nuestra versión avanzada del algoritmo genético (propuesto) ofrece soluciones al problema de investigación dentro de un rango de optimalidad aceptable y en tiempos de ejecución computacionales que hacen factible su implementación en la práctica operativa de las empresas. Con fundamento en los resultados presentados en el capítulo 9 derivamos las siguientes afirmaciones:
   a. Podemos establecer a un 99% de confiabilidad que el AG propuesto alcanza un porcentaje de optimalidad > ***90%*** en un tiempo computacional ***≤ 3 minutos.***
   b. Podemos establecer a un 93% de confiabilidad que el AG propuesto alcanza un porcentaje de optimalidad > ***92.5%*** en un tiempo computacional ***≤ 5 minutos.***
   c. Podemos establecer a un 100% de confiabilidad que el AG propuesto alcanza un porcentaje de optimalidad > ***92.5%*** en un tiempo computacional ***≤ 8 minutos.***
5. Sin embargo, también debemos establecer que no obstante lo anterior, el algoritmo genético propuesto sólo puede asegurar un 90% de confiabilidad para alcanzar soluciones arriba del 95% de optimalidad y requiriendo tiempos computacionales de hasta ***10 minutos.***
6. Lo anterior nos conduce a establecer que la expectativa de solución del Algoritmo Genético (el propuesto y los otros dos que fueron comparados), se ve sensiblemente afectado en la medida en la cual se requieran soluciones que se acerquen al óptimo verdadero (exacto). Lo anterior aunque era previsible, hasta antes del experimento se desconocía cuantitativamente su efecto.

7. Así, se puede verificar en la tabla 9.2 en la columna "***P(x≥95%)***", la forma en la cual va mermando drásticamente la confiabilidad del algoritmo genético propuesto al momento de ir disminuyendo la cantidad de tiempo computacional ofrecido para la solución. Como evidencia de lo anterior se tiene que:

   Podemos establecer que el algoritmo genético propuesto sólo ofrece un 54% de confiabilidad cuando se requiere alcanzar un porcentaje de optimalidad ***≥95%*** *en un tiempo computacional* ***≤5 minutos****. Lo anterior se traduce en el experimento al poderse alcanzar una optimalidad igual o mayor del 95% solo en 22 instancias de las 40 unidades experimentales que fueron aplicados para un límite de 5 minutos de tiempo computacional.*

VI. En lo referente a la justificación económica de la propuesta algorítmica:

1. El análisis costo beneficio expuesto contribuyó a ofrecer las justificaciones económicas acerca de la viabilidad del algoritmo propuesto para mejorar la capacidad de transporte y el servicio al cliente sobre la base de supuestos conservadores definidos empíricamente por el investigador.
2. De acuerdo al análisis expuesto, la conveniencia económica para la aplicación del algoritmo genético está influenciado básicamente por tres factores:
   d. Economía de escala de la empresa (# de Rutas en operación = EE)
   e. Factor de Rutas a optimizar (optimización de la capacidad de transporte = FR)
   f. Factor de Ventas a incrementar (optimización en el servicio al cliente = FV)
3. Aún en empresas debajo de 50 rutas (EE = 50), resulta recomendable la aplicación del algoritmo. Su contribución se verifica significativamente para valores de FR ≥ 5%. Alternativamente también puede verificarse dicha contribución para valores de FV ≥3% y de una manera mucho más importante cuando ambos factores se conjugan.
4. Del factor (EE), tenemos que para el 2do cuartil, es decir 50 ≤EE ≤100, el máximo beneficio económico sería de $4.9 Millones y el promedio de $2.1 Millones por año.
5. En las empresas de gran escala (4to cuartil: 200 ≤EE ≤500), tenemos que el máximo beneficio económico sería de $26.7 Millones y el promedio de $12.8 Millones por año.

VII. En lo referente al estudio analítico y predictivo de los parámetros del Algoritmo Genético:

1. El grado dimensional de los coeficientes obtenidos, contribuyen a establecer el efecto que cada uno de los parámetros individualmente ocasiona en el diseño del espacio solución.
2. Los resultados del experimento, apuntan a señalar los siguientes parámetros como significativos estadísticamente hablando: tamaño de la población (N), cantidad de generaciones (G) y factor de mutación (FM).
3. Con el diseño experimental desarrollado, se comprueba que el parámetro FC (factor de cruzamiento), no es estadísticamente significativo, lo cual confirma lo concluido por Pongcharoen [156].
4. Los coeficientes de estimación para el modelo de regresión lineal desarrollado, evidencian que aumentar el tamaño de la población (N) favorece positivamente en la obtención de mejores soluciones en el algoritmo genético. Lo mismo puede decirse respecto de la cantidad de generaciones (G), aunque su impacto resulta menor en términos relativos respecto al anterior.
5. Para el coeficiente del factor de mutación (FM), también se verifica que aplicarlo con suficiente intensidad logra obtener mejores resultados en el algoritmo genético. No obstante los intervalos de confianza nos hace prever que éste parámetro debe ser cuidadosamente analizado sobre la base de un modelo de predicción cuadrático.
6. Para establecer conclusiones definitivas acerca de la definición de los niveles óptimos de operación para los parámetros del algoritmo genético propuesto, se recomienda desarrollar diseños experimentales del tipo "esparcido" tales como los propuestos por Stewardson y Hicks[157].

Con las conclusiones anteriores, el investigador establece un balance positivo de los objetivos y alcances definidos en el proyecto de investigación [158].

---

[156] Pongcharoen, P. (2000), *Using genetic algorithms for scheduling the production of capital goods*, Proceedings of the 11th International working seminar on production economics, Austria, pag 441-455.

[157] Supra: revisar pag 265.

[158] Supra: revisar los apartados 7.2 al 7.4

VIII. En lo referente a nuevas líneas de investigación sugeridas:

Se prevé que el desempeño obtenido por ambas propuestas de investigación, tanto la del algoritmo de ramificación y corte (BC) como la del algoritmo genético avanzado puedan servir para generar nuevas líneas de investigación particularmente para atender:

6. Instancias múltiples de problemas de logística de ruteo tales como el VRP-TW y el MPDP-TW.
7. Las instancias "no estáticas" de los problemas de ruteo, las cuales son frecuentes encontrar en las empresas que trabajan en ambientes de despacho dinámico.

En lo que respecta a ésta segunda línea de investigación, podemos afirmar que el tiempo de solución computacional se vuelve aún más crítico, ya que en estos casos no se dispone desde un principio del total de las órdenes a tener que atender a los clientes. Por tanto es requerido tomar en consideración el hecho de que las órdenes de entrega y/o de recolección de producto, van a irse incorporando al programa de ruteo a medida que va transcurriendo la jornada de trabajo lo cual implica lo siguiente:

1. El problema a resolver adquiere una mayor complejidad matemática.
2. Se requiere re-programar al estrategia de distribución más frecuentemente y en la misma medida en la que vayan ingresado en firme las órdenes de atención al cliente.
3. Se requiere un algoritmo que pueda ser ejecutado práctica y frecuentemente y que a la vez pueda ofrecer soluciones razonablemente aceptables (cercanas al óptimo).

Tomando en cuenta los argumentos anteriores, el investigador ratifica la importancia que desde el punto de vista científico y práctico existe al promover una propuesta basada en la aplicación de un algoritmo genético. Los resultados expuestos en el presente proyecto de investigación, hacen prever que el algoritmo genético propuesto se vería aún más favorecido hacia su aplicación al ser empleado para atender problemas de ruteo en ambientes de despacho dinámico. Es comprensible que en dicho escenario el binomio "optimalidad" versus "tiempo computacional" se volverá aún más crítico.

i. Anexo A.

Fase de descomposición basada en el método SPP para el análisis y explotación topológica de la versión original de la red logística.

```
model Tspf2
      uses "mmxprs", "mmsystem"
      declarations
          kregen = 1     ! 1= Uno a la vez, 2 = Limite , 3 = Todos
          klim = 200     ! # de Nodos totales: Reales + Topológicos
          ktol = 0                    !0.001
          kmaxtime = 100
            nodos = 1..klim
          kmotor = XPRS_BAR
      ki,kj,kw,kimay,krest,map,vrest,emap,ect,zcosto: integer
      wlim,ksec,dist,knmay,klen,ksol,emin,hsol: real
      zori,zdest: integer
      ktipo,kori,kdest: array(nodos) of integer
      costo: dynamic array(nodos,nodos) of real
      sol: dynamic array(nodos,nodos) of real
      x: dynamic array(nodos,nodos) of mpvar
      xrest: array(range) of linctr
      cred: dynamic array(nodos,nodos) of real
      end-declarations

      ! VERSION ORIGINAL PARA EL MANEJO DEL TW SENCILLO
      ! INCLUYE 4 FASES DE DEPURACION
      ! XPRS_BAR, XPRS_PRI, XPRS_DUAL

      fopen("TOPO.TXT",F_INPUT)
      readln
      repeat
            readln(zori,zdest,zcosto)
            if zcosto >= costo(zori,zdest) then
                  costo(zori,zdest) := zcosto
                  create(x(zori,zdest))
                  x(zori,zdest) is_binary
            end-if
      until (getparam('nbread') < 3)
      fclose(F_INPUT)

      writeln("PROCESO DE CARGA TERMINADO...")

      starttime := gettime
      finalize(nodos)

      setparam("XPRS_PRESOLVE",0)          ! SIN = 0, CON = 1
      setparam("XPRS_MIPRELSTOP",ktol)     ! TOLERANCIA
      setparam("XPRS_MAXTIME",kmaxtime)    ! # DE SEGUNDOS POR MIP
      !setparam("XPRS_CUTSTRATEGY",1)       ! 1=CONSERV, 2= AGRESIVO
      !setparam("XPRS_MIPPRESOLVE",2)       ! +1=Reduce Cost, +2=Logical Bin,
+4=Probing Bin
      !setparam("XPRS_MAXMIPSOL",20)        ! # DE SOLUCIONES ENTERAS
      !setparam("XPRS_MIPLOG",0)            ! SIN LOG EN MIP
```

```
      !setparam("XPRS_CUTSTRATEGY",-1)        ! CUT AUTOMATICO

      forall(i in nodos) do
      if i <= 100 then
            ktipo(i) := 1
      else
            ktipo(i) := 0
      end-if
      end-do

fopen("RED.TXT",F_OUTPUT)
ki := 101
repeat
      kj := 101
      repeat
            if ki <> kj then
                  writeln ("0", ", ", ki , ", " , kj, ", ", "0")
                  forall(i in nodos) kori(i):= 0
                  forall(i in nodos) kdest(i):= 0
                  kori(ki) := 1
                  kdest(kj) := 1

                  forall(j in nodos)
                        xrest(j) := kori(j) + sum(i in nodos) x(i,j) = sum(h in
nodos) x(j,h) + kdest(j)
                  distancia:= sum(i in nodos, j in nodos)
                                    (costo(i,j)*x(i,j) + x(i,j))
                  minimize(kmotor,distancia)
                  if getprobstat = XPRS_INF then
                        writeln ("error:", ki, ", " , kj)
                        exit(ki)
                  end-if

                  hsol := sum(i in nodos, j in nodos) getsol(x(i,j))
                  ksol := getobjval - hsol
                  forall(i in nodos, j in nodos) sol(i,j):= getsol(x(i,j))

                  zi := ki
                  zj  := 1
                  cred(ki,kj) := ksol
                  repeat
                        if sol(zi,zj) = 1 then
                              writeln("1" , ", ", zi,", ",zj, ", ", costo(zi,zj))
                              sol(zi,zj) := 0
                              zi := zj
                              zj := 0
                        end-if
                        zj := zj + 1
                  until zj > klim
                  forall(j in nodos) xrest(j) -= xrest(j)
            end-if
            kj := kj + 1
      until kj > klim
      ki := ki + 1
until ki >  klim
fclose(F_OUTPUT)
```

```
fopen("CRED.TXT",F_OUTPUT)
    forall(i in 101..klim, j in 101..klim)
        writeln(i, ", ", j, ", " , cred(i,j))
fclose(F_OUTPUT)

end-model
```

ii. Anexo B.

Fase de compresión vía estrategia de clusterización mediante criterios de maximización de afinidad.

```
model Tspf1
     uses "mmxprs", "mmsystem"
     declarations
          kred = 100
        klim = 50
        kbound = 15
        ktol = 0                    !0.001
        kmaxtime = 20
        ptos = 1..kred
          nodos = 1..klim
        knear = 1..kbound
        kmotor = XPRS_BAR
    ki,kj,kw,kimay,krest: integer
    wlim,ksec,dist,knmay,klen: real
    kx,ky,top,tcl: array(nodos) of real
    kdel,kpup: array(nodos) of integer
        kmaxclust = 3
    cdist: array(ptos,ptos) of real
    y: array(ptos,ptos) of mpvar
    z: array(ptos) of mpvar
    zx,zy: array(ptos) of real
    ztop,ztcl,zdel,zpup: array(ptos) of integer
    end-declarations

    ! VERSION ORIGINAL PARA EL MANEJO DEL TW SENCILLO
    ! INCLUYE 4 FASES DE DEPURACION
    ! XPRS_BAR, XPRS_PRI, XPRS_DUAL

    fopen("TSPTW100.TXT",F_INPUT)
    kn:= 1
    repeat
    read(ki,zx(kn),zy(kn),ztop(kn),ztcl(kn),zdel(kn),zpup(kn))
    kn := kn + 1
    until (getparam('nbread') < 7)
    fclose(F_INPUT)
    startime := gettime

    ! ****** INICIA FASE DE CLUSTERIZACION
    forall(i in ptos, j in ptos)
          cdist(i,j) := round(((zx(i) - zx(j))^2 +
                              (zy(i) - zy(j))^2)^0.5)

    forall(i in ptos, j in ptos) y(i,j) is_binary
    forall(j in ptos) z(j) is_binary

    forall(i in ptos) sum(j in ptos) y(i,j) = 1
    forall(j in ptos) sum(i in ptos) y(i,j) <= kmaxclust*z(j)
    sum(j in ptos) z(j) = klim
    cluster:= sum(i in ptos, j in ptos) cdist(i,j)*y(i,j)
```

```
      setparam("XPRS_PRESOLVE",0)             ! SIN = 0, CON = 1
      setparam("XPRS_MIPPRESOLVE",0)          ! +1=Reduce Cost, +2=Logical Bin,
+4=Probing Bin
      setparam("XPRS_CUTSTRATEGY",2)          ! CUT AUTOMATICO
      setparam("XPRS_MAXTIME",kmaxtime)       ! # DE SEGUNDOS POR MIP

      minimize(kmotor,cluster)

      k := 0
      forall(j in ptos) do
            wn := 0
            wx := 0.0
            wy := 0.0
            wop := klim
            wcl := 0
            wdel := 0
            wpup := 0
            forall(i in ptos) do
                  if getsol(y(i,j)) >= 0.999 then
                        wn := wn + 1
                        wx := wx + zx(i)
                        wy := wy + zy(i)
                        wdel := wdel + zdel(i)
                        wpup := wpup + zpup(i)
                        if ztop(i) < wop then
                              wop := ztop(i)
                        end-if
                        if ztcl(i) > wcl then
                              wcl := ztcl(i)
                        end-if
                  end-if
            end-do
            if wn >= 1 then
                  k := k + 1
                  kx(k) := wx / wn
                  ky(k) := wy / wn
                  top(k) := wop
                  tcl(k) := wcl
                  kdel(k) := wdel
                  kpup(k) := wpup
            end-if
      end-do
      writeln ("*** Termina Fase de Clusterización ***")
      writeln ("Segundos de ejecución:", gettime-startime)
      ! ****** TERMINA FASE DE CLUSTERIZACION

      fopen("TSPTW50.TXT",F_OUTPUT)
      forall(i in nodos)
            writeln(i," ",kx(i)," ", ky(i)," ",round(top(i)/2),"
",round(tcl(i)/2)," ",kdel(i)," ",kpup(i))
      fclose(F_OUTPUT)

      fopen("CLUSTER.TXT",F_OUTPUT)
      forall(i in ptos, j in ptos | getsol(y(i,j)) >= 0.999)
            writeln(i," ",j)
```

```
fclose(F_OUTPUT)

end-model
```

iii. Anexo C.

Fase de compresión discriminante de arcos basada en la heurística de los "k" nodos vecinos más cercanos para obtener la versión compacta de la red.

```
model Tspf2
      uses "mmxprs", "mmsystem"
      declarations
          kregen = 3  ! 1= Uno a la vez, 2 = Limite , 3 = Todos
          klim = 50
          kbound = 15
          ktol = 0                    !0.001
          kmaxtime = 100
            nodos = 1..klim
          knear = 1..kbound
          kmotor = XPRS_BAR
      ki,kj,kw,kimay,krest,map,vrest,emap,ect,kexit: integer
      wlim,ksec,dist,knmay,klen,ksol,emin,hsol: real
      knodo: array(knear) of integer
      kdist: array(knear) of real
      kx,ky,top,tcl: array(nodos) of real
      wdel,wpup:array(nodos) of integer
      twoc: dynamic array(nodos,nodos) of boolean
      twdif: dynamic array(nodos,nodos) of integer
      costo: dynamic array(nodos,nodos) of real
      msol,sol: dynamic array(nodos,nodos) of real
      rut3: dynamic array(nodos) of integer
      arco: dynamic array(nodos,nodos) of integer
      x: dynamic array(nodos,nodos) of mpvar
      zrest:array(range) of integer
      wrest,xrest: array(range) of linctr
      inicargo, wcargo: integer
      capcargo,wip: real
      end-declarations

      ! VERSION ORIGINAL PARA EL MANEJO DEL TW SENCILLO
      ! INCLUYE 4 FASES DE DEPURACION
      ! XPRS_BAR, XPRS_PRI, XPRS_DUAL

      fopen("TSPTW50.TXT",F_INPUT)
      kn:= 1
      repeat
      read(ki,kx(kn),ky(kn),top(kn),tcl(kn),wdel(kn),wpup(kn))
      kn := kn + 1
      until (getparam('nbread') < 7)
      fclose(F_INPUT)

      forall(i in nodos) do
            forall(j in nodos) do
                  if j >= top(i) and j <= tcl(i) then
                  twoc(i,j) := true
            else
                  twoc(i,j) := false
            end-if
```

```
      end-do
end-do

kn := 1
repeat
      ktw := 1
      repeat
            wscan := 0
            if not twoc(kn,ktw) then
                  wscan := 1
                  repeat
                        if ktw - wscan >= 1 then
                              xsc1 := ktw - wscan
                        else
                              xsc1 := klim + (ktw - wscan)
                        end-if
                        if ktw + wscan <= klim then
                              xsc2 := ktw + wscan
                        else
                              xsc2 := (ktw + wscan) - klim
                        end-if
                        if twoc(kn,xsc1) or twoc(kn,xsc2) then
                              break
                        end-if
                        wscan := wscan + 1
                   until false
            end-if
            twdif(kn,ktw) := wscan
            ktw := ktw + 1
       until ktw > klim
       kn := kn + 1
 until kn > klim

 ki:= 1
 repeat
       kj:= 1
       forall (i in knear) kdist(i):= 1e10
       repeat
             dist := round(((kx(ki) - kx(kj))^2 +
                       (ky(ki) - ky(kj))^2)^0.5)
             kk:= 1
             knmay:= 0
             kimay:= 0
             repeat
                   if kdist(kk) > knmay then
                         if kdist(kk) = 1e10 then
                               kimay := kk
                               break
                         else
                               knmay := kdist(kk)
                               kimay := kk
                         end-if
                   end-if
                   kk:= kk + 1
              until kk > kbound
              if kimay <> 0 then
```

```
                kdist(kimay) := dist
                knodo(kimay) := kj
            end-if
            kj:= kj + 1
        until kj > klim
        kk:= 1
        repeat
            if ki < knodo(kk) then
                costo(ki,knodo(kk)) := kdist(kk)
            create(x(ki,knodo(kk)))
            x(ki,knodo(kk)) is_binary
          else
                costo(knodo(kk),ki) := kdist(kk)
            create(x(knodo(kk),ki))
            x(knodo(kk),ki) is_binary
            end-if
            kk:= kk + 1
        until kk > kbound
        ki:= ki + 1
until ki > klim

writeln("PROCESO DE CARGA TERMINADO...")
```

iv. Anexo D.

Fase de generación agresiva de cortes aplicada a la versión compacta de la red.

```
model Tspf2
      uses "mmxprs", "mmsystem"
      declarations
          kregen = 3  ! 1= Uno a la vez, 2 = Limite , 3 = Todos
          klim = 50
          kbound = 15
          ktol = 0                    !0.001
          kmaxtime = 100
            nodos = 1..klim
          knear = 1..kbound
          kmotor = XPRS_BAR
      ki,kj,kw,kimay,krest,map,vrest,emap,ect,kexit: integer
      wlim,ksec,dist,knmay,klen,ksol,emin,hsol: real
      knodo: array(knear) of integer
      kdist: array(knear) of real
      kx,ky,top,tcl: array(nodos) of real
      wdel,wpup:array(nodos) of integer
      twoc: dynamic array(nodos,nodos) of boolean
      twdif: dynamic array(nodos,nodos) of integer
      costo: dynamic array(nodos,nodos) of real
      msol,sol: dynamic array(nodos,nodos) of real
      rut3: dynamic array(nodos) of integer
      arco: dynamic array(nodos,nodos) of integer
      x: dynamic array(nodos,nodos) of mpvar
      zrest:array(range) of integer
      wrest,xrest: array(range) of linctr
      inicargo, wcargo: integer
      capcargo,wip: real
      end-declarations

      ! VERSION ORIGINAL PARA EL MANEJO DEL TW SENCILLO
      ! INCLUYE 4 FASES DE DEPURACION
      ! XPRS_BAR, XPRS_PRI, XPRS_DUAL

      fopen("TSPTW50.TXT",F_INPUT)
      kn:= 1
      repeat
      read(ki,kx(kn),ky(kn),top(kn),tcl(kn),wdel(kn),wpup(kn))
      kn := kn + 1
      until (getparam('nbread') < 7)
      fclose(F_INPUT)

      forall(i in nodos) do
            forall(j in nodos) do
                  if j >= top(i) and j <= tcl(i) then
                  twoc(i,j) := true
            else
                  twoc(i,j) := false
            end-if
            end-do
      end-do
```

```
      writeln("PROCESO DE CARGA TERMINADO...")

      starttime := gettime
      finalize(nodos)

      forall(h in nodos)
            sum(i in 1..h-1) x(i,h) + sum(j in h+1..klim) x(h,j) = 2


      distancia:= sum(i in nodos, j in i+1..klim)
                                              costo(i,j)*x(i,j)

      setparam("XPRS_PRESOLVE",0)             ! SIN = 0, CON = 1
      setparam("XPRS_MIPRELSTOP",ktol)        ! TOLERANCIA
      setparam("XPRS_MAXTIME",kmaxtime)       ! # DE SEGUNDOS POR MIP
      !setparam("XPRS_CUTSTRATEGY",1)          ! 1=CONSERV, 2= AGRESIVO
      !setparam("XPRS_MIPPRESOLVE",2)          ! +1=Reduce Cost, +2=Logical Bin,
+4=Probing Bin
      !setparam("XPRS_MAXMIPSOL",20)           ! # DE SOLUCIONES ENTERAS
      setparam("XPRS_MIPLOG",0)               ! SIN LOG EN MIP
      !setparam("XPRS_CUTSTRATEGY",-1)         ! CUT AUTOMATICO



procedure bucout
      ! ***********  IMPRIME EL ARCHIVO DEL TOUR ************
      forall(i in nodos, j in nodos) sol(i,j):= msol(i,j)
      fopen("TOUR.TXT",F_OUTPUT)
      ki  := emap
      kj  := 1
      ktw := 0
      repeat
            if ki<kj then
                  if sol(ki,kj) = 1 then
                        ktw := ktw + 1
                        writeln(ki," ",kj," ",ktw," ",top(kj)," ",tcl(kj))
                        sol(ki,kj) := 0
                        ki := kj
                        kj := 0
                  end-if
            else
                  if sol(kj,ki) = 1 then
                        ktw := ktw + 1
                        writeln(ki," ",kj," ",ktw," ",top(kj)," ",tcl(kj))
                        sol(kj,ki) := 0
                        ki := kj
                        kj := 0
                  end-if
            end-if
            kj := kj + 1
      until kj > klim
      fclose(F_OUTPUT)
end-procedure
```

```
 procedure bucmenor
      if ksol <= emin then
            ect := 1
            emin := ksol
            emap := map
            forall(i in nodos, j in nodos) msol(i,j):= getsol(x(i,j))
            bucout
      else
            if ksol = emin then
                  ect := ect + 1
            end-if
      end-if
 end-procedure


 procedure bucborra
      zc := 1
      repeat
            if zrest(zc) = 1 then
                  wrest(zc) -= wrest(zc)
                  zrest(zc) := 0
                  vrest := vrest - 1
                  if kregen = 1 then
                        break
                  else
                        if kregen = 2 then
                              if vrest <= 20 then
                                    break
                              end-if
                        end-if
                  end-if
            end-if
            zc := zc + 1
      until zc > krest
end-procedure


procedure buc1
      kc :=1
      repeat
            minimize(kmotor,distancia)
            if getprobstat = XPRS_INF then
                  break
            end-if
            ksol := getobjval
            savebasis(1)

            forall(i in nodos, j in nodos) sol(i,j):= getsol(x(i,j))

            kmay := 0
            kmen := klim
            kr := 0
            repeat
                  forall(i in nodos) rut3(i) := 0

                  ki:= 1
```

```
kp1:=0
repeat
      kj:= 1
      repeat
            if sol(ki,kj) = 1 then
                  kp1:=ki
                  break 2
            end-if
            kj := kj + 1
      until kj > klim
      ki := ki + 1
until ki > klim

if kp1 = 0 then
      break       ! se acabaron los sub-tours
end-if

kp1 := 1
rut3(ki) := 1
kj := 1
kr := kr + 1
repeat
      if ki<kj then
            if sol(ki,kj) = 1 then
                  kp1 := kp1 + 1
                  rut3(kj):= 1
                  sol(ki,kj) := 0
                  ki := kj
                  kj := 0
            end-if
      else
            if sol(kj,ki) = 1 then
                  kp1 := kp1 + 1
                  rut3(kj):= 1
                  sol(kj,ki) := 0
                  ki := kj
                  kj := 0
            end-if
      end-if
      kj := kj + 1
until kj > klim
kp1 := kp1 - 1

if kp1 <= kmen and kp1 >= 2 then
      kmen := kp1
end-if
if kp1 >= kmay then
      kmay:= kp1
end-if

if kp1 = klim then
      kz := 1     ! entra al time window
      mft := klim
      repeat
            forall(i in nodos, j in nodos) sol(i,j):=
getsol(x(i,j))
```

```
                        ki  := kz
                        kj  := 1
                        wok := 1
                        ktw := 1
                        ft := 0
                        klen := 0
                        repeat
                              if ki<kj then
                                    if sol(ki,kj) = 1 then
                                          if not twoc(kj,ktw) then
                                                wok := 0
                                                ft := ft + 1 ! fuera de Time
Windows en kj
                                                kdif := twdif(kj,ktw)
                                                if kdif > klen then
                                                      wj := kj
                                                      wtw := ktw
                                                      klen := kdif
                                                end-if
                                          end-if
                                          ktw := ktw + 1
                                          sol(ki,kj) := 0
                                          ki := kj
                                          kj := 0
                                    end-if
                              else
                                    if sol(kj,ki) = 1 then
                                          if not twoc(kj,ktw) then
                                                wok := 0
                                                ft := ft + 1 ! fuera de Time
Windows en kj
                                                kdif := twdif(kj,ktw)
                                                if kdif > klen then
                                                      wj := kj
                                                      wtw := ktw
                                                      klen := kdif
                                                end-if
                                          end-if

                                          ktw := ktw + 1
                                          sol(kj,ki) := 0
                                          ki := kj
                                          kj := 0
                                    end-if
                              end-if
                              kj := kj + 1
                        until kj > klim
                        if ft < mft then
                              mft := ft          ! menor cantidad de
violaciones al TW
                              map := kz          ! Nodo de inicio
                              kap := wj          ! Nodo con la mayor violación
de TW
                              ztw := wtw         ! Hr de Llegada del Nodo con
mayor violación de TW
                        end-if
```

```
                        if wok = 1 then
                              kc := kc + 1
                              writeln (ksol, " = ", kmen, ",", kmay, ",", kr,
" Ti= ", gettime-starttime)
                              break 3
                        end-if
                        kz := kz + 1
                  until kz > klim
                  ! A CONTINUACION SE AGREGA EL CUT DE LA RUTA GLOBAL
                  forall(i in nodos, j in nodos) sol(i,j):= getsol(x(i,j))
                  sum(k1 in nodos, k2 in nodos |
                        sol(k1,k2) = 1 and k1<k2 )
                        x(k1,k2) <= kp1-1

                  ! A CONTINUACION SE IMPRIME LA MEJOR SECUENCIA DEL TOUR
                  forall(i in nodos, j in nodos) arco(i,j):= 0
                  forall(i in nodos, j in nodos) sol(i,j):= getsol(x(i,j))
                  ki  := map
                  kj  := 1
                  ktw := 1
                  repeat
                        if ki<kj then
                              if sol(ki,kj) = 1 then

                                    if ktw<>1 and ktw<>klim then
                                          if twoc(kap,ktw+1) and
twoc(kap,ktw-1) then
                                                if kap < kj then
                                                      arco(kap,kj) := 1
                                                else
                                                      arco(kj,kap) := 1
                                                end-if
                                          end-if
                                    end-if
                                    sol(ki,kj) := 0
                                    ktw := ktw + 1
                                    ki := kj
                                    kj := 0
                              end-if
                        else
                              if sol(kj,ki) = 1 then
                                    if ktw<>1 and ktw<>klim then
                                          if twoc(kap,ktw+1) and
twoc(kap,ktw-1) then
                                                if kap < kj then
                                                      arco(kap,kj) := 1
                                                else
                                                      arco(kj,kap) := 1
                                                end-if
                                          end-if
                                    end-if
                                    sol(kj,ki) := 0
                                    ktw := ktw + 1
                                    ki := kj
                                    kj := 0
                              end-if
```

```
                        end-if
                        kj := kj + 1
                    until kj > klim

                    ! A CONTINUACION SE AGREGA EL SUB-CUT PARCIAL
                    krest := krest + 1
                    zrest(krest) := 1
                    wrest(krest) := sum(k1 in nodos, k2 in nodos |
                                            arco(k1,k2) = 1 )
                                                x(k1,k2) >= 1
                    vrest := vrest + 1
                    writeln ("sale del TW con: ", mft, " Rest= ", vrest)
                    writeln(kap,",",ztw,",", top(kap),",",tcl(kap))

                end-if

                if kp1 < klim then
                    ! SE AGREGA EL SUB-TOUR ELIMINATION CONSTR
                    sum(k1 in nodos, k2 in nodos |
                            rut3(k1) = 1 and rut3(k2) = 1 and k1<k2 )
                              x(k1,k2) <= kp1-1
                end-if
            until kp1 <= 0 or kp1 >= klim

            loadprob(distancia)
            loadbasis(1)

            kc := kc + 1
            writeln (ksol, " = ", kmen, ",", kmay, ",", kr, " Ti= ", gettime-
starttime)
            if ksol > 1.05*hsol then
                bucborra
            end-if
        until false
        hsol := ksol
end-procedure


!*************************************************************************
!***** PROCEDIMIENTO PARA LA CONSTRUCCION DE LOS CUTS PARA EL PDP  ********
!*************************************************************************
procedure bucpdp
    kc :=1
    repeat
        minimize(kmotor,distancia)
        if getprobstat = XPRS_INF then
            writeln ("*** E R R O R ***")
            break
        end-if
        ksol := getobjval
        savebasis(1)
        forall(i in nodos, j in nodos) sol(i,j):= getsol(x(i,j))

        kmay := 0
        kmen := klim
        kr := 0
```

```
repeat
      forall(i in nodos) rut3(i) := 0
      ki:= 1
      kp1:=0
      repeat
            kj:= 1
            repeat
                  if sol(ki,kj) = 1 then
                        kp1:=ki
                        break 2
                  end-if
                  kj := kj + 1
            until kj > klim
            ki := ki + 1
      until ki > klim

      if kp1 = 0 then
            break       ! se acabaron los sub-tours
      end-if

      kp1 := 1
      rut3(ki) := 1
      kj := 1
      kr := kr + 1
      repeat
            if ki<kj then
                  if sol(ki,kj) = 1 then
                        kp1 := kp1 + 1
                        rut3(kj):= 1
                        sol(ki,kj) := 0
                        ki := kj
                        kj := 0
                  end-if
            else
                  if sol(kj,ki) = 1 then
                        kp1 := kp1 + 1
                        rut3(kj):= 1
                        sol(kj,ki) := 0
                        ki := kj
                        kj := 0
                  end-if
            end-if
            kj := kj + 1
      until kj > klim
      kp1 := kp1 - 1

      if kp1 <= kmen and kp1 >= 2 then
            kmen := kp1
      end-if
      if kp1 >= kmay then
            kmay:= kp1
      end-if

      if kp1 = klim then
            kz := 1     ! Entra al procedimiento de cuts para el PDP
            mft := klim
```

```
                        repeat
                              forall(i in nodos, j in nodos) sol(i,j):=
getsol(x(i,j))
                              ki  := kz
                              kj  := 1
                              wok := 1
                              ktw := inicargo
                              ft := 0
                              klen := 0
                              repeat
                                    if ki<kj then
                                          if sol(ki,kj) = 1 then
                                                ktw := ktw - wdel(kj) + wpup(kj)
                                                if ktw > capcargo then
                                                      wok := 0
                                                      ft := ft + 1 ! Excede
capacidad de carga en kj
                                                      kdif := ktw - capcargo
                                                      if kdif > klen then
                                                            wj := kj
                                                            wtw := ktw
                                                            klen := kdif
                                                      end-if
                                                end-if
                                                sol(ki,kj) := 0
                                                ki := kj
                                                kj := 0
                                          end-if
                                    else
                                          if sol(kj,ki) = 1 then
                                                ktw := ktw - wdel(kj) + wpup(kj)
                                                if ktw > capcargo then
                                                      wok := 0
                                                      ft := ft + 1 ! Excede
capacidad de carga en kj
                                                      kdif := ktw - capcargo
                                                      if kdif > klen then
                                                            wj := kj
                                                            wtw := ktw
                                                            klen := kdif
                                                      end-if
                                                end-if
                                                sol(kj,ki) := 0
                                                ki := kj
                                                kj := 0
                                          end-if
                                    end-if
                                    kj := kj + 1
                              until kj > klim
                              if ft < mft then
                                    mft := ft   ! menor cantidad de violaciones al
PDP
                                    map := kz   ! Apuntador al Nodo de inicio
                                    kap := wj   ! Nodo con la mayor violación del
PDP
```

```
                              ztw := wtw  ! Carga del Nodo con mayor
violación al PDP
                        end-if
                        if wok = 1 then
                              map := kz
                              kc := kc + 1
                              writeln (ksol, " = ", kmen, ",", kmay, ",", kr,
" Ti= ", gettime-starttime)
                              break 3
                        end-if
                        kz := kz + 1
                  until kz > klim
                  ! A CONTINUACION SE AGREGA EL CUT DE LA RUTA GLOBAL
                  forall(i in nodos, j in nodos) sol(i,j):= getsol(x(i,j))
                  sum(k1 in nodos, k2 in nodos |
                        sol(k1,k2) = 1 and k1<k2 )
                        x(k1,k2) <= kp1-1

                  ! SE BUSCAN LAS SEC'S HACIA DONDE IR A UBICAR EL EXCESO DE
CARGA
                  forall(i in nodos, j in nodos) arco(i,j):= 0
                  forall(i in nodos, j in nodos) sol(i,j):= getsol(x(i,j))
                  ki  := map
                  kj  := 1
                  ktw := inicargo
                  repeat
                        if ki<kj then
                              if sol(ki,kj) = 1 then
                                    ktw := ktw - wdel(kj) + wpup(kj)
                                    if ktw - wdel(kap) + wpup(kap) < capcargo
then
                                          if kap < kj then
                                                arco(kap,kj) := 1
                                          else
                                                arco(kj,kap) := 1
                                          end-if
                                    end-if
                                    sol(ki,kj) := 0
                                    ki := kj
                                    kj := 0
                              end-if
                        else
                              if sol(kj,ki) = 1 then
                                    ktw := ktw - wdel(kj) + wpup(kj)
                                    if ktw - wdel(kap) + wpup(kap) < capcargo
then
                                          if kap < kj then
                                                arco(kap,kj) := 1
                                          else
                                                arco(kj,kap) := 1
                                          end-if
                                    end-if
                                    sol(kj,ki) := 0
                                    ki := kj
                                    kj := 0
                              end-if
```

```
                        end-if
                        kj := kj + 1
                    until kj > klim

                    ! A CONTINUACION SE AGREGA EL SUB-CUT PARCIAL
                    krest := krest + 1
                    zrest(krest) := 1
                    wrest(krest) := sum(k1 in nodos, k2 in nodos |
                                        arco(k1,k2) = 1 )
                                            x(k1,k2) >= 1
                    vrest := vrest + 1
                    writeln ("sale del PDP con: ", mft, " Rest= ", vrest)
                    writeln(kap,",",ztw,",", capcargo)
                end-if

                if kp1 < klim then
                    ! SE AGREGA EL SUB-TOUR ELIMINATION CONSTR
                    sum(k1 in nodos, k2 in nodos |
                            rut3(k1) = 1 and rut3(k2) = 1 and k1<k2 )
                              x(k1,k2) <= kp1-1
                end-if
            until kp1 <= 0 or kp1 >= klim

            loadprob(distancia)
            loadbasis(1)

            kc := kc + 1
            writeln (ksol, " = ", kmen, ",", kmay, ",", kr, " Ti= ", gettime-
starttime)
        until false
        hsol := ksol
        if getprobstat <> XPRS_INF then
            bucmenor
        end-if
end-procedure



procedure buc2
    ! ***********  INICIA DEPURACION ************
if krest > 0 and vrest > 0 then
    writeln ("*** Depuración:", ksol, " = ", vrest, " Best= ",emin)
    forall(i in 1..krest) xrest(i) := wrest(i)

    kc:= 1
    repeat
        if zrest(kc) = 1 then
            wrest(kc) -= wrest(kc)
            loadprob(distancia)
            loadbasis(1)
            minimize(kmotor,distancia)
            ksol := getobjval
            savebasis(1)
            kz := 1     ! entra al Time Window y al PDP
            repeat
                forall(i in nodos, j in nodos) sol(i,j):= getsol(x(i,j))
```

```
                        ki  := kz
                        kj  := 1
                        wok := 1
                        ktw := 0
                        wcargo := inicargo
                        repeat
                              if ki<kj then
                                    if sol(ki,kj) = 1 then
                                          ktw := ktw + 1
                                          wcargo := wcargo - wdel(kj) + wpup(kj)
                                          if not twoc(kj,ktw) or wcargo > capcargo
then
                                                wok := 0
                                                break
                                          end-if
                                          sol(ki,kj) := 0
                                          ki := kj
                                          kj := 0
                                    end-if
                              else
                                    if sol(kj,ki) = 1 then
                                          ktw := ktw + 1
                                          wcargo := wcargo - wdel(kj) + wpup(kj)
                                          if not twoc(kj,ktw) or wcargo > capcargo
then
                                                wok := 0
                                                break
                                          end-if
                                          sol(kj,ki) := 0
                                          ki := kj
                                          kj := 0
                                    end-if
                              end-if
                              kj := kj + 1
                        until kj > klim
                        if wok = 1 then
                              map := kz
                              break
                        end-if
                        kz := kz + 1
                  until kz > klim
                  if wok = 1 and ktw >= klim then
                        vrest := vrest - 1
                        writeln("F1 Borra:", kc, " = ", ksol, " Rest= ", vrest, "
Best= ",emin)
                        zrest(kc) := 0
                        bucmenor
                  else
                        wrest(kc) += xrest(kc)
                  end-if
            end-if
            kc := kc + 1
      until kc > krest

! ***********  FASE INTERMEDIA DE DEPURACION ************
repeat
```

```
zsig := 0
kc:= 1
repeat
      if zrest(kc) = 1 then
            ztime := gettime
            wrest(kc) -= wrest(kc)
            loadprob(distancia)
            loadbasis(1)
            minimize(kmotor,distancia)
            ksol := getobjval
            savebasis(1)
            kcic := 0
            repeat
                  forall(i in nodos, j in nodos) sol(i,j):= getsol(x(i,j))
                  kmay := 0
                  kmen := klim
                  kr := 0
                  repeat
                        forall(i in nodos) rut3(i) := 0
                        ki:= 1
                        kp1:=0
                        repeat
                              kj:= 1
                              repeat
                                    if sol(ki,kj) = 1 then
                                          kp1:=ki
                                          break 2
                                    end-if
                                    kj := kj + 1
                              until kj > klim
                              ki := ki + 1
                        until ki > klim

                        if kp1 = 0 then
                              break       ! se acabaron los sub-tours
                        end-if

                        kp1 := 1
                        rut3(ki) := 1
                        kj := 1
                        kr := kr + 1
                        repeat
                              if ki<kj then
                                    if sol(ki,kj) = 1 then
                                          kp1 := kp1 + 1
                                          rut3(kj):= 1
                                          sol(ki,kj) := 0
                                          ki := kj
                                          kj := 0
                                    end-if
                              else
                                    if sol(kj,ki) = 1 then
                                          kp1 := kp1 + 1
                                          rut3(kj):= 1
                                          sol(kj,ki) := 0
                                          ki := kj
```

```
                        kj := 0
                    end-if
                end-if
                kj := kj + 1
            until kj > klim
            kp1 := kp1 - 1

            if kp1 = klim then
                kz := 1     ! entra al time window y al PDP
                repeat
                    forall(i in nodos, j in nodos) sol(i,j):=
getsol(x(i,j))
                    ki  := kz
                    kj  := 1
                    wok := 1
                    ktw := 0
                    wcargo := inicargo
                    repeat
                        if ki<kj then
                            if sol(ki,kj) = 1 then
                                ktw := ktw + 1
                                wcargo := wcargo -
wdel(kj) + wpup(kj)
                                if not twoc(kj,ktw) or
wcargo > capcargo then
                                    wok := 0
                                    break
                                end-if
                                sol(ki,kj) := 0
                                ki := kj
                                kj := 0
                            end-if
                        else
                            if sol(kj,ki) = 1 then
                                ktw := ktw + 1
                                wcargo := wcargo -
wdel(kj) + wpup(kj)
                                if not twoc(kj,ktw) or
wcargo > capcargo then
                                    wok := 0
                                    break
                                end-if

                                sol(kj,ki) := 0
                                ki := kj
                                kj := 0
                            end-if
                        end-if
                        kj := kj + 1
                    until kj > klim
                    if wok = 1 then
                        map := kz
                        break 3
                    end-if
                    kz := kz + 1
                until kz > klim
```

```
                                    ! SE AGREGA EL CUT DE LA RUTA GLOBAL
                                    forall(i in nodos, j in nodos) sol(i,j):=
getsol(x(i,j))
                                    sum(k1 in nodos, k2 in nodos |
                                          sol(k1,k2) = 1 and k1<k2 )
                                            x(k1,k2) <= kp1-1
                              end-if
                              if kp1 < klim then
                                    ! SE AGREGA EL SUB-TOUR ELIMINATION CONSTR
                                    sum(k1 in nodos, k2 in nodos |
                                          rut3(k1) = 1 and rut3(k2) = 1 and
k1<k2 )
                                            x(k1,k2) <= kp1-1
                              end-if
                        until kp1 <= 0 or kp1 >= klim
                        loadprob(distancia)
                        loadbasis(1)
                        minimize(kmotor,distancia)
                        ksol := getobjval
                        savebasis(1)
                        kcic := kcic + 1
                  until kp1 >= klim
                  if wok = 0 then
                     . wrest(kc) += xrest(kc)
                  else
                        vrest := vrest - 1
                        writeln("F2 Borra:", kc, " = ", ksol, " Rest= ", vrest, "
Best= ", emin)
                        zrest(kc) := 0
                        zsig := 1
                        bucmenor
                  end-if
            end-if
            kc := kc + 1
      until kc > krest
      ! ***********  TERMINA FASE INTERMEDIA DE DEPURACION ************
until zsig = 0 or vrest <= 0
end-if
end-procedure



! ***********  PROCEDIMIENTO PRINCIPAL ************
emin := 1e6
ect := 0
krest := 0
vrest := 0
zfirst := 0
hsol := 1e6
kexit := 0
inicargo := sum(i in nodos) wdel(i)
capcargo := maxlist(sum(i in nodos) wdel(i), sum(1 in nodos) wpup(i))*1.9
writeln ("DEL:",sum(i in nodos) wdel(i), " PUP:",sum(i in nodos) wpup(i), "
CAP:",capcargo)

repeat
```

```
      buc1
      bucpdp
      if getprobstat = XPRS_INF then
            bucborra
      else
            buc2
            if vrest = 0 then
                  kexit := 1
            else
                  bucborra
            end-if
      end-if
until gettime-starttime > 100000 or kexit = 1

! ***********  FINALIZA EJECUCION DEL MODELO ************
      ksol := emin
      forall(i in nodos, j in nodos) sol(i,j):= msol(i,j)
      writeln ("*** Termina:", ksol)

      ki  := emap
      kj  := 1
      ktw := 0
      wcargo := inicargo
      wip := 0
      repeat
            if ki<kj then
                  if sol(ki,kj) = 1 then
                        ktw := ktw + 1
                        wcargo := wcargo - wdel(kj) + wpup(kj)
                        wip := wip + wcargo
                        writeln(ki,",",kj,",",ktw,",",top(kj),",",tcl(kj)," W=
",wdel(kj), ", ", wpup(kj), ", ", wcargo)
                        sol(ki,kj) := 0
                        ki := kj
                        kj := 0
                  end-if
            else
                  if sol(kj,ki) = 1 then
                        ktw := ktw + 1
                        wcargo := wcargo - wdel(kj) + wpup(kj)
                        wip := wip + wcargo
                        writeln(ki,",",kj,",",ktw,",",top(kj),",",tcl(kj)," W=
",wdel(kj), ", ", wpup(kj), ", ", wcargo)
                        sol(kj,ki) := 0
                        ki := kj
                        kj := 0
                  end-if
            end-if
            kj := kj + 1
      until kj > klim
      writeln("Tiempo: ",gettime-starttime)
      wip := wip / 50
      writeln("WIP: ",wip)

end-model
```

v. Anexo E.

Fase evolutiva generacional para explotar el "pool" de cortes generados.

```
Sub mipexa()

Dim i, j, k, n As Integer
Dim kmen, kref As Double
Dim karcos, kxy, kagrupa, kinforef, kapert, kcierre As Variant

SolverOk SetCell:="$B$198", MaxMinVal:=2, ValueOf:=0, ByChange:="Arcos,Clusters",
Engine _
    :=4, EngineDesc:="XPRESS LP/MIP Solver"
SolverSolve (True)

kxy = Application.Names("theLocations").RefersTo
karcos = Application.Names("arcos").RefersTo
kagrupa = Application.Names("agrupa").RefersTo
kinforef = Application.Names("inforef").RefersTo
kapert = Application.Names("apertura").RefersTo
kcierre = Application.Names("cierre").RefersTo

Range(kinforef).Select
Selection.ClearContents

n = Range(karcos).Rows.Count
kref = 0

k = 0
For j = 1 To n
    h = 0
    x = 0
    y = 0
    kent = 1000#
    ksal = 0
    For i = 1 To n
        If Range(karcos).Rows(i).Columns(j).Value >= 0.999 Then
            If h = 0 Then
                k = k + 1
                h = h + 1
                Range(kinforef).Rows(k).Columns(h).Value = k
            End If
            h = h + 1
            Range(kinforef).Rows(k).Columns(h).Value = i
            x = x + Range(kxy).Rows(i).Columns(1)
            y = y + Range(kxy).Rows(i).Columns(2)
            If Range(kapert).Rows(i).Value < kent Then
                kent = Range(kapert).Rows(i).Value
            End If
            If Range(kcierre).Rows(i).Value > ksal Then
                ksal = Range(kcierre).Rows(i).Value
            End If
        End If
    Next i
    If x <> 0 Then
```

```
        Range(kinforef).Rows(k).Columns(5).Value = x / (h - 1)
        Range(kinforef).Rows(k).Columns(6).Value = y / (h - 1)
        Range(kinforef).Rows(k).Columns(7).Value = kent
        Range(kinforef).Rows(k).Columns(8).Value = ksal
        Range(kinforef).Rows(k).Columns(9).Value = h - 1
    End If
Next j


'*****************************************************************************
'***** SE EJECUTA EL PROCEDIMIETNO DE MIRUTA
Sub miruta()

Dim i, j, k, kord, n, kp, kt, kp1, kp2, kpaso, ki As Integer
Dim ktrial, kmen, kdist1, kdist2 As Double
Dim RangeOrder As Variant
Dim ksec(100), ktour(100), ktourx(100), ktoury(100) As Integer
Dim ksigue As Boolean

RangeOrder = Application.Names("theOrder").RefersTo
n = Range(RangeOrder).Rows.Count

ksigue = True
i = 1
kord = 1
Range(RangeOrder).Rows(kord) = i
ktour(kord) = i
ksec(i) = 9999

While ksigue
    kmen = 999999
    ksigue = False
    For kp = 1 To kord
        For j = 1 To n
            If ksec(j) <> 9999 Then
                ktrial = midistancia(ktour(kp), j)
                If ktrial < kmen Then
                    kmen = ktrial
                    kt = kp
                    k = j
                    ksigue = True
                End If
            End If
        Next j
    Next kp
    If ksigue = True Then
        kord = kord + 1
        Range(RangeOrder).Rows(kord) = k
        ktour(kord) = k
        ksec(k) = 9999
        i = k
    End If
    If kord >= 4 Then
        kp1 = kt - 1
        kp2 = kt + 1
        If kt = 1 Then
```

```
            kp1 = kord - 1
        End If
        If kt = kord - 1 Then
            kp2 = 1
        End If

'*************************************************************************************
        If kt = 1 Then
            For ki = 1 To kord
                ktourx(ki) = ktour(ki)
            Next ki
        End If
        If kt = kord - 1 Then
            For ki = 1 To kp1
                ktourx(ki) = ktour(ki)
            Next ki
            ktourx(kord - 1) = ktour(kord)
            ktourx(kord) = ktour(kord - 1)
        End If

        If kt <> 1 And kt <> kord - 1 Then
            For ki = 1 To kp1
                ktourx(ki) = ktour(ki)
            Next ki
            ktourx(kt) = ktour(kord)
            For ki = kp2 To kord
                ktourx(ki) = ktour(ki - 1)
            Next ki
        End If
'*************************************************************************************
        If kt = kord - 1 Then
            For ki = 1 To kord
                ktoury(ki) = ktour(ki)
            Next ki
        End If
        If kt = 1 Then
            ktoury(1) = ktour(1)
            ktoury(2) = ktour(kord)
            For ki = 3 To kord
                ktoury(ki) = ktour(ki - 1)
            Next ki
        End If
        If kt <> 1 And kt <> kord - 1 Then
            For ki = 1 To kt
                ktoury(ki) = ktour(ki)
            Next ki
            ktoury(kp2) = ktour(kord)
            For ki = kp2 + 1 To kord
                ktoury(ki) = ktour(ki - 1)
            Next ki
        End If
'*************************************************************************************
        kdist1 = 0
        For ki = 1 To kord - 1
            kdist1 = kdist1 + midistancia(ktourx(ki), ktourx(ki + 1))
        Next ki
```

```
        kdist1 = kdist1 + midistancia(ktourx(kord), ktourx(1))

        kdist2 = 0
        For ki = 1 To kord - 1
            kdist2 = kdist2 + midistancia(ktoury(ki), ktoury(ki + 1))
        Next ki
        kdist2 = kdist2 + midistancia(ktoury(kord), ktoury(1))

        If kdist1 <= kdist2 Then
            For ki = 1 To kord
                ktour(ki) = ktourx(ki)
                Range(RangeOrder).Rows(ki) = ktour(ki)
            Next ki
        Else
            For ki = 1 To kord
                ktour(ki) = ktoury(ki)
                Range(RangeOrder).Rows(ki) = ktour(ki)
            Next ki
         End If
    End If
Wend
'*****************************************************************************************

k2apert = Application.Names("ajapert").RefersTo
k2cierre = Application.Names("ajcierre").RefersTo

n2 = Range(k2apert).Rows.Count


For m = 1 To n2
    For i = 1 To k
        For j = 1 To 3
            If Range(kinforef).Rows(i).Columns(j + 1).Value = m Then
                Range(k2apert).Rows(m).Value =
                            Range(kinforef).Rows(i).Columns(7).Value
                Range(k2cierre).Rows(m).Value =
                            Range(kinforef).Rows(i).Columns(8).Value
                j = 100
                i = 100
            End If
        Next j
    Next i
Next m


'*****************************************************************************************
'***** SE EJECUTA EL PROCEDIMIETNO DE SECFIN2
secfin2
'*****************************************************************************************
Sub secfin2()

Dim i, j, k, nord, nsec, ki As Integer
Dim p(3) As Integer
Dim kvl(100) As Integer
Dim kmen, kref As Double
Dim kvar(100) As String
```

```
Dim korder, ksecfin, kagrupa, ksecref As Variant
Dim kcom1 As String

Sheets("INP").Select

korder = Application.Names("theorder").RefersTo
kagrupa = Application.Names("agrupa").RefersTo
ksecfin = Application.Names("secfin").RefersTo

Range(ksecfin).Select
Selection.ClearContents

nord = Range(korder).Rows.Count
nsec = Range(ksecfin).Rows.Count

kcolj = 1
kcolv = 6
kv = 1
For i = 1 To nord
    ki = Range(korder).Rows(i).Value
    p(1) = Range(kagrupa).Rows(ki).Columns(1).Value
    p(2) = Range(kagrupa).Rows(ki).Columns(2).Value
    p(3) = Range(kagrupa).Rows(ki).Columns(3).Value
    klen = Sgn(p(1)) + Sgn(p(2)) + Sgn(p(3))
    If klen = 1 Then
        Range(ksecfin).Rows(kcolj).Value = p(1)
        kcolj = kcolj + 1
    Else
        kini = kcolv
        kvar(kv) = "U" & kini & ":U" & kini + klen - 1
        kvl(kv) = klen
        kv = kv + 1
        For k = 1 To klen
            kcom2 = "=V" & kcolv
            Range(ksecfin).Rows(kcolj).Formula = kcom2
            kcom1 = "U" & kcolv
            Range(kcom1).Value = k
            kcom1 = "V" & kcolv
            kcom2 = "=INDEX(W" & kini & ":W" & kini + klen - 1 & ",U" & kcolv &
",1)"
            Range(kcom1).Formula = kcom2
            kcom1 = "W" & kcolv
            Range(kcom1).Value = p(k)
            kcolj = kcolj + 1
            kcolv = kcolv + 1
        Next k
    End If
Next i

SolverReset

kcom1 = "U6:U" & kcolv - 1
SolverAdd cellRef:=kcom1, relation:=3, formulaText:="$BA$1"

For i = 1 To kv - 1
    If kvl(i) = 2 Then
```

```
        kcom1 = "$BA$2"
    Else
        kcom1 = "$BA$3"
    End If
    SolverAdd cellRef:=kvar(i), relation:=1, formulaText:=kcom1
    SolverAdd cellRef:=kvar(i), relation:=4
    SolverAdd cellRef:=kvar(i), relation:=6
Next i

SolverEVOptions MaxTime:=10000, Iterations:=100000, Precision:=0.000001, _
        Convergence:=0.0001, PopulationSize:=0, MutationRate:=0.075,
RequireBounds:= _
        True, StepThru:=False, Scaling:=False, AssumeNonneg:=True, BypassReports:=
_
        False, LocalSearch:=4
SolverLimOptions MaxSubProblems:=500000, MaxFeasibleSols:=500000, Tolerance:=0.05,
_
        MaxTimeNoImp:=100, SolveWithout:=False

kcom1 = "U6:U" & kcolv - 1
SolverOk SetCell:="$R$16", MaxMinVal:=2, ValueOf:=0, ByChange:=kcom1, Engine _
    :=3, EngineDesc:="Standard Evolutionary"

SolverSolve (True)
End Sub
```

vi. Anexo F.

Fase de descompresión de la ruta propuesta para la versión original de la red

```
model Tspf3
      uses "mmxprs", "mmsystem"
      declarations
            kred = 100
          klim = 50
          kbound = 10
          ktol = 0                    !0.001
          kmaxtime = 100
          ptos = 1..kred
            nodos = 1..klim
          knear = 1..kbound
          kmotor = XPRS_BAR
      ki,kj,kw,kimay,krest: integer
      wlim,ksec,dist,knmay,klen: real
      kx,ky,top,tcl: array(nodos) of real
      sol: dynamic array(ptos,ptos) of real
          kmaxclust = 3
      cdist: array(ptos,ptos) of real
      zx,zy: array(ptos) of real
      ztop,ztcl: array(ptos) of integer
      zclus: array(ptos) of integer
      zgroup: array(ptos) of integer
      w: dynamic array(ptos,ptos) of mpvar
      trinp,trout,trsec,trot,trct:array(nodos) of integer
      rut3: dynamic array(ptos) of integer
      arco: dynamic array(ptos,ptos) of integer
      end-declarations

      ! VERSION ORIGINAL PARA EL MANEJO DEL TW SENCILLO
      ! INCLUYE 4 FASES DE DEPURACION
      ! XPRS_BAR, XPRS_PRI, XPRS_DUAL

      startime := gettime

      fopen("TSPTW100.TXT",F_INPUT)
      kn:= 1
      repeat
      read(ki,zx(kn),zy(kn),ztop(kn),ztcl(kn))
      kn := kn + 1
      until (getparam('nbread') < 5)
      fclose(F_INPUT)

      forall(i in ptos, k in ptos) do
            if i <> k then
                  cdist(i,k) := round(((zx(i) - zx(k))^2 +
                                    (zy(i) - zy(k))^2)^0.5)
            else
                  cdist(i,k) := 1e6
            end-if
      end-do
```

```
fopen("CLUSTER.TXT",F_INPUT)
repeat
      read(kn,zclus(kn))
      if getparam('nbread') >= 2 then
      zgroup(zclus(kn)) := zgroup(zclus(kn)) + 1
end-if
until (getparam('nbread') < 2)
fclose(F_INPUT)

fopen("TOUR.TXT",F_INPUT)
kn := 1
repeat
      read(trinp(kn),trout(kn),trsec(kn),trot(kn),trct(kn))
      kn := kn + 1
until (getparam('nbread') < 5)
fclose(F_INPUT)

kn := 0
forall(j in ptos) do
      if zgroup(j) >= 1 then
            kn:= kn + 1
            ztrad(kn) := j
      end-if
end-do

forall(j in nodos) do
      ar1 := ztrad(trinp(j))
      ar2 := ztrad(trout(j))
      forall(i in ptos, k in ptos) do
            if  (zclus(i) = ar1 and zclus(k) = ar1) or
                  (zclus(i) = ar1 and zclus(k) = ar2)  then
                  create(w(i,k))
                  w(i,k) is_binary
            end-if
      end-do
end-do

forall(i in ptos) sum(j in ptos) w(i,j) = 1
forall(j in ptos) sum(i in ptos) w(i,j) = 1
distancia:= sum(i in ptos, j in ptos) cdist(i,j)*w(i,j)


! INICIA PROCESO PARA ELIMINACION DE SUB-TOURS
kc :=1
repeat
      minimize(kmotor,distancia)
      ksol := getobjval
      savebasis(1)

      forall(i in ptos, j in ptos) sol(i,j):= getsol(w(i,j))

      kmay := 0
      kmen := kred
      kr := 0
      repeat
            forall(i in ptos) rut3(i) := 0
```

```
   ki:= 1
   kp1:=0
   repeat
         kj:= 1
         repeat
               if sol(ki,kj) = 1 then
                     kp1:=ki
                     break 2
               end-if
               kj := kj + 1
         until kj > kred
         ki := ki + 1
    until ki > kred

    if kp1 = 0 then
          break       ! se acabaron los sub-tours
    end-if

    kp1 := 1
    rut3(ki) := 1
    kj := 1
    kr := kr + 1
    repeat
          if sol(ki,kj) = 1 then
                kp1 := kp1 + 1
                rut3(kj):= 1
                sol(ki,kj) := 0
                ki := kj
                kj := 0
          end-if
          kj := kj + 1
     until kj > kred
     kp1 := kp1 - 1

     if kp1 <= kmen and kp1 >= 2 then
           kmen := kp1
     end-if
     if kp1 >= kmay then
           kmay:= kp1
     end-if

     if kp1 < kred then
           ! SE AGREGA EL SUB-TOUR ELIMINATION CONSTR
           sum(k1 in ptos, k2 in ptos |
                     rut3(k1) = 1 and rut3(k2) = 1 )
                       w(k1,k2) <= kp1-1
      end-if
until kp1 <= 0 or kp1 >= kred
if kp1 = kred then
      break
end-if
loadprob(distancia)
loadbasis(1)

kc := kc + 1
```

```
            writeln (ksol, " = ", kmen, ",", kmay, ",", kr, " Ti= ", gettime-
startime)
      until false

      minimize(kmotor,distancia)
      ksol := getobjval
      writeln ("*** Termina:", ksol)

      forall(i in ptos, j in ptos) sol(i,j):= getsol(w(i,j))
      ki  := ztrad(trinp(1))
      kj  := 1
      ktw := 0
      repeat
            if sol(ki,kj) <> 0 then
                  ktw := ktw + 1
                  writeln(ki,",",kj,",",ktw,",",ztop(kj),",",ztcl(kj))
                  sol(ki,kj) := 0
                  ki := kj
                  kj := 0
            end-if
            kj := kj + 1
      until kj > kred
      writeln("Tiempo: ",gettime-startime)


      fopen("RUTA.TXT",F_OUTPUT)
      forall(i in ptos, j in ptos) sol(i,j):= getsol(w(i,j))
      ki  := ztrad(trinp(1))
      kj  := 1
      ktw := 0
      repeat
            if sol(ki,kj) <> 0 then
                  ktw := ktw + 1
                  writeln(ki,",",kj,",",ktw,",",ztop(kj),",",ztcl(kj))
                  sol(ki,kj) := 0
                  ki := kj
                  kj := 0
            end-if
            kj := kj + 1
      until kj > kred
      fclose(F_OUTPUT)

end-model
```

12. Referencias Bibliográficas.

Aguerrondo, Inés. (1998), *La Educación del tercer milenio*, "Revista Academia", Argentina.

Alander, Jarmo. (2003), *An Indexed Bibliography of Genetic Algorithms in Logistics*, Department of Engineering, University of Vaasa, Finland.

Applegate, D; Bixby, R; Chvátal, V; (1998), *On the solution of traveling salesman problems*. "Documenta Mathematica Extra Volume ICM III", EUA.

Applegate, David. (1995), *Finding Cuts in the TSP*, Princeton University, EUA.

Arsham, Hossein. (1996), *Applied Management Science*, Cambridge University Press, EUA.

Ascheuer, N; Fischetti, M; Grotschel, M. (2001), *Solving ATSP with time windows by branch-and-cut*, Springer-Verlag, EUA.

Ascheuer, N; Jünger, M; Reinelt, G. (2000), *A branch & cut algorithm for the asymmetric Traveling Salesman Problem with precedence constraints*, "Computational Optimization and Applications 17(1)", EUA.

Baker, Kenneth. (1999), *Gaining Insight in Linear Programming*, "Informs", Vol.1, EUA.

Balas, E; Simonetti, N. (1996), *Linear time dynamic programming algorithms for some new classes of restricted TSP's* , "Management Science Research Report 617", Graduate School of Industrial Administration, Carnegie Mellon University, EUA.

Bednarek, A. (1984), *Analogies between Analogies: The mathematical reports of S.M. Ulam and his Los Alamos collaborators*, University of California Press, EUA.

Bellman, Richard. (1957), Dynamic Programming, Princeton University Press, Princeton Nueva Jersey.

Berry, William. (1990), *Management Decision Science*, 1thEd, Irwin, EUA.

Bixby, Robert. (1999), *MIP: theory and practice closing the gap*, Ilog Cplex Division, Department of Computational and Applied Mathematics Rice University, Houston EUA.

Bremermann, H. (1962), *Optimization through Evolution and Recombination*, Spartan Books, Alemania.

Bryne, John. (1997), *Los Nuevos gurúes de la Administración*, "Revista Gestión y Estrategia", No.11-12, México.

Buckles, Bill. (1992), *Genetic Algorithms*, IEEE Computer Society Press, EUA.

Cheng, R. (1997), *Genetic Algorithms and Engineering Design*, John Wiley Interscience, Boston Ma, EUA.

Chowdury, M. (1997), *Messy Genetic Algorithms based new learning method for fussy controllers*, "Proceedings of the IEEE International Conference on Industrial Technology", The MIT Press, Shangai China.

Christofides, N. (1976), *Worst-case analysis of a new heuristic for the traveling salesman problem*, "Report 388 Graduate School of Industrial Administration", Carnegie-Mellon University, Pittsburgh EUA.

Cochran, W. (1977), *Experimental Design*, John Wiley & Sons, New York.

Coello, Carlos. (1995), *Introducción a los Algoritmos Genéticos*, "Tecnologías de Información y Estrategias de Negocios, No. 17", México.

Cook, W; Rich, Jennifer. (1999), *A parallel cutting-plane algorithm for the vehicle routing problem with time windows*, Computational and Applied Mathematics Rice University, Houston EUA.

Dale, Mann. (1999), *Achievement gains from a Statewide Comprehensive Intructional Program*, Milken Familiy Foundation, EUA.

Dávila, S. (1999), *El papel del docente en la calidad educativa*, México, en http://www.nalejandria.com/akademeia/sdavila/

Deb, K. (1995), *Simulated Binary crossover in evolutionary optimization*, "Complex Systems No. 9", EUA.

Delgado, Mónica. (2001), *SEP difunde resultados de la evaluación Internacional: Entrevista al Secretario de Educación Reyes Taméz*, "Periódico el Norte", México.

Dell 'Amico, M; Martello, S. (1997), *Linear assignment: Annotated Bibliographies in Combinatorial Optimization*, Wiley & Sons editors, EUA.

Desrosiers, J; Dumas, Y; Solomon, M; Soumis, F. (1995), *Time constrained routing and scheduling*, G.L. Nemhauser editor, Handbooks in Operations Research and Management Science, Holanda.

Drezner, Zvi. (1996), *Facility Location: A Survey of Applications and Methods*, 1st Ed, Springer Verlag, EUA.

Dumas, Y; Desrosiers, J; Gelinas, E; Solomon, M. (1995), *An algorithm for the traveling salesman problem with time windows*, "Operations Research 43(2)", EUA.

Eijl Van, C. (1995), *A polyhedral approach to the delivery man problem*, "Technical Report 95–19", Department of Mathematics and Computer Science, Eindhoven University of Technology, The Neatherlands.

Encuesta Nacional de Empleo. (2001), *Base de Datos y Tabulados de la Muestra Censal*, "XII Censo General de Población y Vivienda", México.

Esbensen, H. (1995), *Computing near-optimal solutions to the Steiner problem using genetics algorithms*, Networks Interscience, Vol 26, No.4, EUA.

Fisher, Marshall. (1995), *Overview over optimization models in transportation*, Handbooks in Operations Research and Management Science, North Holland, EUA.

Fisher, R. (1971), *The Design of Experiments*, Hafner Press & Macmillan Publishers, London.

Focacci, F; Milano, M. (1999), *Solving Tsp with time windows with constraints*, "INFORMS Journal of Computing, ICLP 99", International Conference on Logic Programming, EUA.

Focacci, Filippo; Milano, Michela. (2000), *Solving TSP with Time Windows with Constraints*, Dip. Ingegneria, University of Ferrara, Italia.

Fuente Guía Roji. (2000), *Censo de empresas del área metropolitana de la ciudad de Monterrey*, México.

Garey, D. (1979), *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, New York.

Garey, M; Johnson, D. (1989), *Computers and Intractability: a Guide to the Theory of NP-Completeness*, Freeman Ed, San Francisco EUA.

Geoffrion, Arthur. (1999), *The decadence of operations research*, ORMS Today, EUA.

Geoffrion, Arthur. (2000), *The value of operations research*, "ORMS Today", EUA.

Glover, Fred. (1986), *Future paths for integer programming and links to artificial intelligence*, "Computers and Operations Research", EUA.

Glover, Fred. (1990), *Tabu Search: A Tutorial*, "Interfaces, Vol 20, No. 4", EUA.

Glover, Fred. (1993), *A user's guide to Tabu Search*, "Annals of Operations Research, Vol. 41", EUA.

Goldberg, D. (1995), *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley Pub Co, University of Massachusetts, EUA.

Goldberg, D. (1996), *Don't' worry, be messy*, "Proceedings of the 9th International Conference on Genetic Algorithms", The MIT Press, Cambridge Mass, EUA.

Goldberg, David. (1989), *Genetic Algorithms in Optimization and Machine Learning*, Addison-Wesley Publishing Company, EUA.

Gomez, M. (1998), *Inserción ocupacional de los egresados universitarios*, "IV Congreso Nacional de Estudios del Trabajo", Argentina.

Gomory, R. (1963), *An algorithm for integer solutions to linear programs*, Recent Advances in

Mathematical Programming (R. L. Graves and P. Wolfe, eds.), McGraw-Hill, New York.
Greenberg , H. (1996), *Mathematical Programming Glossary*, disponible en http://www.cudenver.edu/~hgreenbe/glossary.

Hartmann, Sonke. (2000), *Project Scheduling Under Limited Resources: Models, Methods, and Applications*, Springer Verlag, EUA.

Hicks, C. (1994), *Fundamental Concepts in the Design of Experiments*, Holt and Winston Ed, New York.

Hinkelmann, K. (1994), *Design and Analysis of Experiments*, McMillan, EUA.

Hoffmeister, F. (1992), *Genetic Algorithms and Evolution Strategies: Similarities and Differences*, "Technical Report No.1", University of Dortmund, Alemania.

Holland, John. (1975), *Adaptation in Natural and Artificial Systems*, University of Michigan Press, EUA.

Hollstein, R. (1991), *Artificial Genetic adaptation in computer control systems*, Phd Thesis, University of Michigan, EUA.

Hsiang, Thomas. (2001), *The Illusion of Power*, "ORMS Today", EUA.
Hunter, W. (1986), Statistics for Experimenters, John Willey & Sons, EUA.

Jonker, R; Volgenant, T. (1982), *A branch and bound algorithm for the symmetric traveling salesman problem*, "European Journal of Operational Research, No. 2", Dinamarca.

Jünger, M; Rinaldi, G; Thienel, S. (2000), *Practical performance of efficient minimum cut algorithms*, "Algorithmica No. 26", Dinamarca.

Karger, D. (1993), *Global min-cuts in RNC and other ramifications of a simple min-cut algorithm*, "Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms", EUA.
Klein, Robert. (1999), *Scheduling of Resource-Constrained Projects*, Kluwer Academic, EUA.

Koza, John. (1992), *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, The MIT Press, EUA.

Laguna, Manuel. (1994), *A guide to implementing Tabu Search*, "Technical Report, Graduate School of Business, University of Colorado", Boulder Colorado.

Land, A; Doig, H. (1960), *An Automatic Method for Solving Discrete Programming Problems*, Econometrica, EUA.

Lee, R. (1976), *Worst-case analysis of a new heuristic for the travelling salesman problem*, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh PA. Disponible en http://www.ms.uky.edu/~jlee/jlsup/jlsup.html.

Lenstra, J. (1985), *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Ediitorial Wiley, Chichester England.

Lenstra, K. (1990), *A Variable Depth Approach for the Single-Vehicle Pickup and Delivery Problem with Time Windows*, "COSOR No. 90-48", Eindhoven University of Technology, Alemania.

Levine, G. (1997), *Developing fitter genetic algorithms*, INFORMS Journal of Computing, Vol. 9, No. 5, EUA.

Lorenzen, T. (1993), *Design of Experiments*, Marcel Dekker Inc, EUA.

Marchand, Horacio. (2001), *¿Una nación en Riesgo?*, EGADE ITESM, México.

Mathias, K. (1994), *Transforming the search space with gray coding*, Proceedings of the IEEE International Conference on Evolution Computation, Piscataway New Jersey, EUA.

Mingozzii, A; Bianco, L; Ricciardelli, S. (1997), *Dynamic programming strategies for the travelling salesman problem with time windows and precedence constraints*, "Operations Research No. 45", EUA.

Mitrovic, Snezana. (1998), *Pickup and Delivery Problem with Time Windows*, "Technical Report SFU CMPT TR 1998-12", Canada.

Mittelmann, H. (2002), *Decision Tree for Optimization Software*, disponible en http://plato.la.asu.edu/guide.html.

Montgomery, D. (1991), *Diseño y análisis de experimentos*, Grupo Editorial Ibero-Américo, México.

Moore, Lawrance. (1993), *Management Science*, 4thEd, Allyn and Bacon, EUA.

Mota, Flavio. (1998), *El deber ser y el querer ser en la Educación*, "Revista Academia", México, disponible en http:://kepler.uag.mx/temasedu/deberser.htm.

Optima. (1998), *Mathematical Programming Society Newletter*, EUA.

Padberg, M; Rinaldi, G. (1991), *A Branch and cut algorithm for the resolution of large-scale symmetric traveling salesman problem*, "SIAM Review No. 33", EUA.

Palmgren, Myrna. (2001), *A Column Generation Algorithm for the Log Truck Scheduling Problem*, Department of Science and Technology (ITN), Linköping University, Norrköping Sweden.

Parker, R; Rardin, R. (1998), *Discrete Optimization*, Academic Press, New York EUA.
Pawda, Juan. (2000), *Modelos de Investigación de Operaciones*, 1era.Ed, Limusa, México.

Premkumar, G. (1999), *Telecomunications Network design: a genetic algorithm approach*, School of Information Sciences and Technology & Pennsylvania State College of Business Administration, EUA.

Quintana, Enrique. (2001), *Matemáticas y Economía*, "Editorial el Norte", México.

Render, Barry. (2000), *Quantitative Analysis for Management*, 7thEd, Prentice Hall, EUA.

Reeves, C. (1995), *Genetics algorithms flow shop sequencing*, Computers & Operations Research, Vol 22, EUA.

Ríos, Roger. (1999), *Aplicaciones del TSP*, "Ingenierías 2(4)", México.

Rodríguez, Joaquin. (1998), *Introd. a la Administración con Enfoque de Sistemas*, 3era Ed, ECAFSA, México.

Savelsberg, M. (1995), *Local search in Routing Problem with Time Windows*, Annals of Operations Research, Rotherdam Holanda.

Savelsbergh, M. (1998), *Local Search in Physical Distribution Management*, Eindhoven University of Technology, Rottherdam Holanda.

Skiena, S. (1990), *Implementing Discrete Mathematics: Combinatorics and Graph Theory in Mathematica*, Addison-Wesley, Redwood CA. Disponible en ftp://ftp.cs.sunysb.edu/pub/Combinatorica/

Sodhi, ManMohan. (2001), *A Match Made in Heaven*, "ORMS Today", EUA.

Solomon, M. (1984), *On the worst-case performance of some heuristics for the vehicle routing and scheduling problem with time window constraints*, "Report 83-05-03", The Wharton School, University of Pennsylvania EUA.

Srinivas, M. (1999), *Genetic Algorithms: A Survey*, IEEE Computer, EUA.

Stewardson, D. (2002), *Overcoming Complexity: optimizing genetic algorithms for use in complex scheduling problems via designed experiments*, University of Newcastle, United Kingdom.

Streifel, R. (1999), *Dynamic Fussy control of genetic algorithm parameters*, IEEE Trans. Syst., EUA.

Tenti, Emilio. (2000), *Educación Participativa*, Argentina, disponible en http://www.utdt.edu/eduforum/.

Toutenggburg, H. (1995), *Experimental Design and Model Choice*, Physica-Verlag, EUA.

Tsitsiklis, J. (1992), *Special cases of traveling salesman and repairman problems with time windows*, "Networks No. 22", EUA.

Turing, A. (1950), *Computer Machinery and Intelligence*, MIND, EUA.

Weber, D. (2000), *A First Course in the Design of Experiments*, CRC Press, EUA.

Whitley, D. (1998), *Representation issues in Neighborhood Search and Evolutionary Algorithms*, John Wiley and Sons, University of Sussex, England, Capítulo 3.

Quién domina el Tiempo, domina el Espacio.

J. Fabián López P.