Concorde code：

<http://www.math.uwaterloo.ca/tsp/concorde/downloads/downloads.htm>

<http://www.codeforge.cn/read/182415/3OPT.C__html>

/\*

        HEADER:         CUG000.05;

        TITLE:          ThreeOpt;

        DATE:           Mar 89;

        DESCRIPTION:    3-Opt Tour Improvement Algorithm;

        VERSION:        2.0;

        FILENAME:       3OPT.C;

        SEE-ALSO:       TSP.C, NN.C, POPT.C, 2OPT.C, HYBRID.C, FN.C,

                        BOOLEAN.H, NODELIST.H, TSP.H,

                        BLDMTX.C, PRTMTX.C, TIME.C;

        AUTHORS:        Kevin E. Knauss;

\*/

#include "nodelist.h"

long ThreeOpt (NumberOfVirteces, Path)

   unsigned NumberOfVirteces;

   NODE \*Path;

{

   unsigned ArcCost();

   long CircuitCost;

   unsigned count, index, pindex, sindex, j, k;

   unsigned D1, D2, D3, D4;

   NODE \*First, \*Last, \*Kth, \*Jth, \*Save, \*Reverse;

   count = 1;

   First = Path;

   do {

      Last = First->prior;

      Kth = First->next;

      /\* when j = k+1, D1 checks k=1 case (i.e. single point) \*/

      do {

         Jth = Kth->next;

         /\* when j = k+1, D2 checks 2-Opt \*/

         do {

            D1 = ArcCost (Kth->position, Jth->next->position)

               + ArcCost (First->position, Jth->position);

            D2 = ArcCost (First->position, Jth->next->position)

               + ArcCost (Kth->position, Jth->position);

            D3 = ArcCost (Kth->next->position, Last->position);

            D4 = ArcCost (First->position, Last->position)

               + ArcCost (Kth->position, Kth->next->position)

               + ArcCost (Jth->position, Jth->next->position);

            if (((D1 + D3) < D4) || ((D2 + D3) < D4)) {

               Last->next = Kth->next;

               Kth->next->prior = Last;

               if (D1 <= D2) {

                  Kth->next = Jth->next;

                  Kth->next->prior = Kth;

                  Jth->next = First;

                  First->prior = Jth;

} else {

                  for ( Reverse = First; Reverse != Kth; Reverse = Save) {

                     Save = Reverse->next;

                     Reverse->next = Reverse->prior;

                     Reverse->prior = Save;

}

                  First->next = Jth->next;

                  Jth->next->prior = First;

                  Kth->next = Kth->prior;

                  Kth->prior = Jth;

                  Jth->next = Kth;

}

               count = 0;

               First = Last->next;

               Kth = First->next;

               Jth = Kth->next;

} else

               Jth = Jth->next;

} while ((Jth != Last->prior) && (count != 0));

         Kth = Kth->next;

} while ((Kth != Last->prior->prior->prior) && (count != 0));

      if (count != 0)

         First = First->next;

      count++;

} while (count < NumberOfVirteces);

   Last = First->prior;

   CircuitCost = ArcCost (Last->position, First->position);

   for ( Kth = First; Kth != Last; Kth = Kth->next)

      CircuitCost += ArcCost (Kth->position, Kth->next->position);

   return (CircuitCost);

} /\* ThreeOpt \*/