# Transport Solver

Manual de Usuario

David Jiménez Xolalpa A00994415

Alejandro Méndez González A01210989

**Introducción:**

El método de transporte sirve para poder encontrar la mejor ruta de transporte de productos desde diferentes puntos de destino a diferentes puntos de origen, de acuerdo a la demanda y oferta entre los diferentes lugares. Te dice cuanto enviar por cada ruta.

**Algoritmo en Pseudocódigo:**

*Algoritmo general*

1. Obtener la tabla de transporte
2. Encontrar la solución inicial
3. Calcular índices de mejoramiento
    a. Si todos son positivos, ir a fin.
4. Mejorar solución
5. Ir a 3
6. FIN

*Solución inicial: Esquina noroeste*

1. Posicionarse en la esquina superior derecha de la tabla
2. Dar la mayor cantidad posible a la celda actual, del origen al destino correspondiente, dentro de los límites de cada lugar
3. Si la demanda del destino se satisface pero la oferta sobra, moverse a la derecha
4. Si no se cubre la demanda pero se llega al máximo de la oferta, moverse hacia abajo
5. Si ambos se cumplen, moverse en diagonal abajo a la derecha
6. Si no se cumple con la demanda ni la oferta, ir a 2.
7. FIN

*Stepping Stone (para cada ciclo)*

1. Ir a la primer celda desocupada
2. Marcar la celda como temporalmente ocupada
3. Eliminar todas las filas que cuenten con una sola celda ocupada
4. Eliminar todas las columnas que cuenten con una sola celda ocupada
5. Si se eliminó alguna fila o columna, ir a 3
6. Buscar horizontalmente una celda ocupada no visitada
7. Si hubo celda ocupada, ir a 9
8. Buscar verticalmente una celda ocupada no visitada
9. Moverse a la celda ocupada
10. Marcar la celda como visitada
11. Si la celda actual no es la celda temporalmente ocupada, ir a 6
12. FIN

**Código Fuente:**

```
// ************ Transporte ***************


function Crea() { }

Crea.prototype.ejemplo = function() {

    var fila1 = new Array(5);
    fila1[0] = "De/a";
    fila1[1] = "Albuquerque";
    fila1[2] = "Boston";
    fila1[3] = "Cleveland";
    fila1[4] = "Oferta";

    var fila2 = new Array(5);
    fila2[0] = "Des Moines";
    fila2[1] = 5;
```

```
    fila2[2] = 4;
    fila2[3] = 3;
    fila2[4] = 100;

    var fila3 = new Array(5);
    fila3[0] = "Evansville";
    fila3[1] = 8;
    fila3[2] = 4;
    fila3[3] = 3;
    fila3[4] = 300;

    var fila4 = new Array(5);
    fila4[0] = "Fort Lauderdale";
    fila4[1] = 9;
    fila4[2] = 7;
    fila4[3] = 5;
    fila4[4] = 300;

    var fila5 = new Array(5);
    fila5[0] = "Demanda";
    fila5[1] = 300;
    fila5[2] = 200;
    fila5[3] = 200;
    fila5[4] = 700;

    this.tabla = new Array(5);
    this.tabla[0] = fila1;
    this.tabla[1] = fila2;
    this.tabla[2] = fila3;
    this.tabla[3] = fila4;
    this.tabla[4] = fila5;

};

Crea.prototype.tabla = null; //this.tabla this table holds the costs,
supply & demand information
Crea.prototype.tabla2 = null;//this.tabla this table holds the "units
to be transported" information
Crea.prototype.tablaDummy  =  null;//auxiliary  table  to  find  the
stepping stone loops

Crea.current = null;

Crea.prototype.htmlString = "";
```

```
//initializes the tabla2 array with the same dimensions
Crea.prototype.inicializaTabla2 = function(){
    this.tabla2 = new Array(this.tabla.length);
    for(var i = 0; i < this.tabla2.length;i++){
        this.tabla2[i] = new Array(this.tabla[i].length);
    }
};

//this method helps print the problem information in a table
Crea.prototype.imprime = function(noRes) {
    var    html    =    "<table    width=200    border=1    cellpadding=1
cellspacing=1>";
    for (var i=0;i<this.tabla.length;i++){
      for (var j=0;j<this.tabla[i].length;j++){
            if(i == 0 || i == this.tabla.length-1){
                if(j==0 || j == this.tabla.length-1)
                    html += "<td>" + this.tabla[i][j] + "</td>";
                else
                    html  +=  "<td  colspan=2>"  +  this.tabla[i][j]  +
"</td>";
            } else {
                if(j==0 || j == this.tabla[i].length-1)
                    html += "<td>" + this.tabla[i][j] + "</td>";
                else{
                    var     linea     =     "<td     colspan=2><Table
border=1><tr><td>" + this.tabla[i][j] + "</td></tr></Table>";
                    if(this.tabla2[i][j] == undefined)
                        linea = linea + " </td>";
                    else
                        linea = linea + this.tabla2[i][j] + "</td>";
                    html += linea;
                }
            }
      }
      html += "</tr>";
    }
    html += "</table>";
    html += "</br>";
    if (!noRes) html += this.imprimeSolucion();
    return html;
};

//prints the optimal value of the current problem information table
Crea.prototype.imprimeSolucion = function(){
    var resS = "<p>";
```

```
    var res = 0;
    var tmp = 0;
    for (var i=1;i<this.tabla.length-1;i++){
     for (var j=1;j<this.tabla[i].length-1;j++){
            if(this.tabla2[i][j] != undefined){
                tmp = this.tabla[i][j] * this.tabla2[i][j];
                if(i == this.tabla.length-2){
                    resS += tmp + "=";
                    res += tmp;
                } else {
                    resS += tmp + "+";
                    res += tmp;
                }
            }
        }
    }

    resS += res + "</p></br>";

    return resS;
};

//Calculates the first feasible solution of the problem
//with the Northwest Corner method
Crea.prototype.northwestCorner = function(){
    var i = 1;
    var j = 1;
    while(!this.checaTabla()){
        var     r     =     this.tabla[i][this.tabla[i].length-1]     -
this.sumaFila(i);
        var     c     =     this.tabla[this.tabla.length-1][j]     -
this.sumaColumna(j);
        if(r<c){
            this.tabla2[i][j] = r;
        }else{
            this.tabla2[i][j] = c;
        }
        r = this.tabla[i][this.tabla[i].length-1] - this.sumaFila(i);
        c = this.tabla[this.tabla.length-1][j] - this.sumaColumna(j);
        if(r==0 && c==0){
            if(i != this.tabla.length-2){
                this.tabla2[i+1][j] = 0;
                i++;
                j++;
            }
```

```
        } else if (r==0)
            i++;
        else
            j++;
    }
};

//checks that the units in the tabla2 table don't exceed the supply or
demand restrictions
Crea.prototype.checaTabla = function(){
    var flag = true;

    for (var i =1; i<this.tabla2.length-1; i++){
        if(this.tabla[i][this.tabla[i].length-1] - this.sumaFila(i) !=
0)
            flag = false;
    }

    for (i =1; i<this.tabla2[1].length-1; i++){
        if(this.tabla[this.tabla.length-1][i] - this.sumaColumna(i) !=
0)
            flag = false;
    }
    return flag;
};

//adds the values of the indexCol column in tabla2
Crea.prototype.sumaColumna = function(indexCol){
    var suma = 0;
    for (var i =1; i<this.tabla2.length-1; i++){
        if(this.tabla2[i][indexCol]!=undefined)
            suma += this.tabla2[i][indexCol];
    }
    //alert("SumaCol" + indexCol + ":" + suma);//////quitar
    return suma;
};

//adds the values of the indexFila row in tabla2
Crea.prototype.sumaFila = function(indexFila){
    var suma = 0;
    for (var i =1; i<this.tabla2[indexFila].length-1; i++){
        if(this.tabla2[indexFila][i]!=undefined)
            suma += this.tabla2[indexFila][i];
    }
    //alert("SumaFila" + indexFila + ":" + suma);//////quitar
```

```
    return suma;
};

//calculates the reduced cost indexes
//improving the current solution if a negative index remains
Crea.prototype.steppingStone = function (){
    var indices = null;
    do{
        indices = new Array();
        this.calculaIndices(indices);
        if(this.indicesNegativos(indices)){
            this.mejoraSolucion(indices);
            this.html+= this.imprime();
        }
    } while(this.indicesNegativos(indices));
};

//calculates the reduce cost indexes of all the vacant cells in the
tabla2 table
Crea.prototype.calculaIndices = function (indices) {
    for (var i=1;i<this.tabla2.length-1;i++){
        for (var j=1;j<this.tabla2[i].length-1;j++){
            if (this.tabla2[i][j] == undefined)
                indices.push(this.calculaIndice(i,j));
        }
    }
};

//calculates the reduce cost indexes of a single vacant cell in the
tabla2 table
Crea.prototype.calculaIndice = function (fila, columna){
    var indice = new Array();
    var ciclo = new Array();

    indice = [0,fila,columna];
    var vecinos = new Array();
    vecinos.push([fila,columna]);
    var iteraciones = 0;
    this.creaDummy(fila, columna);
    ciclo = this.buscaCiclo(vecinos, iteraciones);
    ciclo.pop();//quita repetido
    indice[0] = this.calculaCiclo(ciclo);
    return indice;
};
```

```
//finds the coordinates of the cells in tabla2, that forms a stepping
stone loop for the vacant cell stored
//in the vecinos array
Crea.prototype.buscaCiclo = function (vecinos, iteraciones){
    if(!this.repetidos(vecinos)){
        if(this.vecinosFila(vecinos,      vecinos[vecinos.length-1][0],
iteraciones)){
            var      tmp      =      this.encuentraVecinoFila(vecinos,
vecinos[vecinos.length-1][0], iteraciones);
            vecinos.push(tmp);
            iteraciones++;
            vecinos = this.buscaCiclo(vecinos, iteraciones);
        }
        if(!this.repetidos(vecinos)){
            if(this.vecinosColumna(vecinos,     vecinos[vecinos.length-
1][1], iteraciones)){
                var    tmp2    =    this.encuentraVecinoColumna(vecinos,
vecinos[vecinos.length-1][1], iteraciones);
                vecinos.push(tmp2);
                iteraciones++;
                vecinos = this.buscaCiclo(vecinos, iteraciones);
            } else {
                vecinos.pop();
                iteraciones--;
            }
        }
    }
    return vecinos;
};

//creates a dummy table containing only the cells in tabla2, that
forms a stepping stone loop for the vacant cell
//with coordinates (fila,columna)
Crea.prototype.creaDummy = function (fila, columna){
    var i = 0;
    var j = 0;
    this.tablaDummy = new Array(this.tabla2.length);
    for(i = 0; i < this.tablaDummy.length;i++){
        this.tablaDummy[i] = new Array(this.tabla2[i].length);
    }

    for(i = 0; i < this.tabla2.length; i++){
        for(j = 0; j < this.tabla2[i].length;j++){
            this.tablaDummy[i][j] = this.tabla2[i][j];
        }
```

```
    }
    var f = false;
    var c = false;
    do{
        f = this.borraFilasTablaDummy(fila, columna);
        c = this.borraColumnasTablaDummy(fila, columna);
    } while (f || c);
};

//deletes the rows of the dummy table that only have one element
Crea.prototype.borraFilasTablaDummy = function (fila, columna){
    var res = false;
    var i = 0;
    var j = 0;
    var filas = new Array();
    for(i = 0; i < this.tablaDummy.length; i++){
        var contador = 0;
        for(j =0; j < this.tablaDummy[0].length; j++){
            if(  (this.tablaDummy[i][j]!=undefined)  ||  (i==fila  &&
j===columna) ){
                contador++;
            }
        }
        if(contador === 1)
            filas.push(i);
    }

    if(filas.length > 0){
        for(i = 0; i < filas.length; i++){
            for(j =0; j<this.tablaDummy[filas[i]].length; j++){
                this.tablaDummy[filas[i]][j] = undefined;
            }
        }
        res = true;
    }
    return res;
};

//deletes the columns of the dummy table that only have one element
Crea.prototype.borraColumnasTablaDummy = function (fila, columna){
    var res = false;
    var i = 0;
    var j = 0;
    var columnas = new Array();
    for(i = 0; i < this.tablaDummy[0].length; i++){
```

```
        var contador = 0;
        for(j =0; j < this.tablaDummy.length; j++){
            if(  (this.tablaDummy[j][i]!=undefined)  ||  (j==fila  &&
i===columna) ){
                contador++;
            }
        }
        if(contador === 1)
            columnas.push(i);
    }

    if(columnas.length > 0){
        for(i = 0; i < columnas.length; i++){
            for(j =0; j<this.tablaDummy.length; j++){
                this.tablaDummy[j][columnas[i]] = undefined;
            }
        }
        res = true;
    }
    return res;
};

//checks if there is a value in the indexFila row of the tabla2 to
continue
//the path of a stepping stone loop
Crea.prototype.vecinosFila    =    function    (vecinos,    indexFila,
iteraciones){
    var flag = false;
    var i = 0;
    var tmp = new Array();
    if(iteraciones < 3){
        for (i =1; i<this.tablaDummy[indexFila].length-1; i++){
            if(this.tablaDummy[indexFila][i]!=undefined){
                tmp = [indexFila,i];
                if(!this.contiene(vecinos, tmp))
                    flag = true;
            }
        }
    } else {
         for (i =1; i<this.tablaDummy[indexFila].length-1; i++){
            tmp = [indexFila,i];
            if(tmp[0]==vecinos[0][0]&&tmp[1]==vecinos[0][1]){
                flag = true;
            } else if(this.tablaDummy[indexFila][i]!=undefined){
                tmp = [indexFila,i];
```

```
                if(!this.contiene(vecinos, tmp))
                    flag = true;
            }
        }
    }
    return flag;
};

//stores the value in the indexFila row of the tabla2 that continues
//the current path of the stepping stone loop in the vecinos array
Crea.prototype.encuentraVecinoFila  =  function  (vecinos,  indexFila,
iteraciones){
    var i = 0;
    var tmp = new Array();
    tmp = [0,0];
    if(iteraciones < 3){
        for (i =1; i<this.tablaDummy[indexFila].length-1; i++){
            if(this.tablaDummy[indexFila][i]!=undefined){
                tmp = [indexFila,i];
                if(!this.contiene(vecinos, tmp))
                    return tmp;
            }
        }
    } else {
         for (i =1; i<this.tablaDummy[indexFila].length-1; i++){
            tmp = [indexFila,i];
            if(tmp[0]==vecinos[0][0]&&tmp[1]==vecinos[0][1]){
                return tmp;
            } else if(this.tablaDummy[indexFila][i]!=undefined){
                tmp = [indexFila,i];
                if(!this.contiene(vecinos, tmp))
                    return tmp;
            }
        }
    }
    return tmp;
};

//checks if there is a value in the indexCol column of the tabla2 to
continue
//the path of a stepping stone loop
Crea.prototype.vecinosColumna   =   function   (vecinos,   indexCol,
iteraciones){
    var flag = false;
    var i = 0;
```

```
    var tmp = new Array();
    if(iteraciones < 3){
        for (i =1; i<this.tablaDummy.length-1; i++){
            if(this.tablaDummy[i][indexCol]!=undefined){
                tmp = [i,indexCol];
                if(!this.contiene(vecinos, tmp))
                    flag = true;
            }
        }
    } else {
        for (i =1; i<this.tablaDummy.length-1; i++){
            tmp = [i,indexCol];
            if(tmp[0]==vecinos[0][0]&&tmp[1]==vecinos[0][1]){
                flag = true;
            }else if(this.tablaDummy[i][indexCol]!=undefined){
                tmp = [i,indexCol];
                if(!this.contiene(vecinos, tmp))
                    flag = true;
            }
        }
    }

    return flag;
};

//stores the value in the indexFila row of the tabla2 that continues
//the current path of the stepping stone loop in the vecinos array
Crea.prototype.encuentraVecinoColumna = function (vecinos, indexCol,
iteraciones){
    var i = 0;
    var tmp = new Array();
    tmp = [0,0];
    if(iteraciones < 3){
        for (i =1; i<this.tablaDummy.length-1; i++){
            if(this.tablaDummy[i][indexCol]!=undefined){
                tmp = [i,indexCol];
                if(!this.contiene(vecinos, tmp))
                    return tmp;
            }
        }
    } else {
        for (i =1; i<this.tablaDummy.length-1; i++){
            tmp = [i,indexCol];
            if(tmp[0]==vecinos[0][0]&&tmp[1]==vecinos[0][1]){
                return tmp;
```

```
            }else if(this.tablaDummy[i][indexCol]!=undefined){
                tmp = [i,indexCol];
                if(!this.contiene(vecinos, tmp))
                    return tmp;
            }
        }
    }

    return tmp;
};

//checks if the array[n][2] a has the element e[2]
Crea.prototype.contiene = function (a, e){
    var res = false;
    for(var i = 0; i < a.length;i++){
        if(a[i][0] == e[0] && a[i][1] == e[1] )
            res = true;
    }
    return res;
};

//checks if the vecinos array has repeated elements
Crea.prototype.repetidos = function (vecinos){
    for(var i=0; i<vecinos.length;i++){
        for(var j=i+1; j<vecinos.length;j++){
            if(vecinos[i][0]==vecinos[j][0]                                 &&
vecinos[i][1]==vecinos[j][1])
                    return true;
        }
    }
    return false;
};

//Calculates the reduce cost index of the loop contained in the ciclo
array
Crea.prototype.calculaCiclo = function (ciclo){
    var suma = true;
    var resultado = 0;
    for(var i = 0; i<ciclo.length;i++){
        if(suma){
            if(this.tabla[ ciclo[i][0] ][ ciclo[i][1] ] != undefined){
                resultado += this.tabla[ ciclo[i][0] ][ ciclo[i][1] ];
                suma = false;
            } else
                suma = false;
```

```
        } else{
            resultado -= this.tabla[ ciclo[i][0] ][ ciclo[i][1] ];
            suma = true;
        }
    }
    return resultado;
};

//improves the current solution with the information in the indices
array
Crea.prototype.mejoraSolucion = function (indices){
    var mejora = this.masNegativo(indices);
    var ciclo = new Array();
    var vecinos = new Array();
    vecinos.push([ mejora[1] , mejora[2] ]);
    var iteraciones = 0;
    this.creaDummy(mejora[1], mejora[2]);
    ciclo = this.buscaCiclo(vecinos, iteraciones);
    ciclo.pop();//quita repetido
    var unidades = this.menorUnidades(ciclo);

    var suma = true;
    for(var i = 0; i<ciclo.length;i++){
        if(suma){
            if(this.tabla2[   ciclo[i][0]   ][   ciclo[i][1]   ]   !=
undefined){
                this.tabla2[   ciclo[i][0]   ][   ciclo[i][1]   ]   =
this.tabla2[ ciclo[i][0] ][ ciclo[i][1] ] + unidades;
                suma = false;
            } else
                this.tabla2[ ciclo[i][0] ][ ciclo[i][1] ] = unidades;
                suma = false;
        } else{
            this.tabla2[ ciclo[i][0] ][ ciclo[i][1] ] = this.tabla2[
ciclo[i][0] ][ ciclo[i][1] ] - unidades;
            if(this.tabla2[ ciclo[i][0] ][ ciclo[i][1] ] == 0)
               this.tabla2[ ciclo[i][0] ][ ciclo[i][1] ] = undefined;
            suma = true;
        }
    }
};

//finds the most negative reduce cost index in the indices array
Crea.prototype.masNegativo = function(indices){
    var tmp = 1;
```

```
    var res = null;
    var i = 0;
    for(i=0; i<indices.length;i++){
        if(indices[i][0] < tmp)
            tmp = indices[i][0];
    }

    for(i=0; i<indices.length;i++){
        if(indices[i][0] == tmp)
            res = indices[i];
    }
    return res;
};

//finds the lowest units value of the most negative reduce cost index
loop ciclo
//to improve the current solution
Crea.prototype.menorUnidades = function(ciclo){
    var unidades = Number.MAX_VALUE;
    var suma = true;
    for(var i = 0; i<ciclo.length;i++){
        if(suma){
            suma = false;
        } else{
            if(this.tabla2[ ciclo[i][0] ][ ciclo[i][1] ] < unidades)
                unidades = this.tabla2[ ciclo[i][0] ][ ciclo[i][1] ];
            suma = true;
        }
    }
    return unidades;
};

//checks if there is still a negative reduce cost index in the indices
array
Crea.prototype.indicesNegativos = function (indices){
    var flag = false;
    for(var i=0; i<indices.length;i++){
        if(indices[i][0] < 0)
            flag = true;
    }
    return flag;
};

//executes the above methods in the correct order to find THE solution
Crea.prototype.run = function() {
```

```
      this.inicializaTabla2();
      this.html = this.imprime(true);
      this.northwestCorner();
      this.html   +=   "<h3>Primera   Solucion   Factible   (Northwest
Corner)</h3>";
      this.html += this.imprime();
      this.steppingStone();

      //Ejecución
      /*

      var ejemplo = new Crea();
      ejemplo.ejemplo();
      ejemplo.inicializaTabla2();

      ejemplo.northwestCorner();
      document.write("Primera Solucion Factible (Northwest Corner)");
      document.write(ejemplo.imprime());
      ejemplo.steppingStone();
      document.write("Solucion Optima");
      document.write(ejemplo.imprime());
      */
};

// ********************** User Interface ***************
function UI() { }

UI.setData = function() {
      var data = new Array();
      var si = Crea.current;
      for (var i = 0, len = si.numOrigenes + 2; i < len; i++) {
            if (i===0) data[i] = UI.getRowData("origen_header");
            else        if        (i===(len-1))        data[i]        =
UI.getRowData("origen_totals", i /*index*/, true /*suma*/);
            else data[i] = UI.getRowData("origen_r" + (i - 1));
      }
      si.tabla = data;
      console.log(data);
};

UI.getRowData = function(id, index, suma) {
      console.log("gettingRowData");
      var r = $("#" + id);
      console.log(r);
```

```
      var si = Crea.current;
      var cols = si.numDestinos + 2;

      var a = new Array(cols);
      console.log(cols);

      r.find("input").each(function(index) {
            var temp = $(this).val();
            var tempInt = parseInt(temp);
            a[index] = isNaN(tempInt)? temp : tempInt;
      });
      return a;
};

UI.showOptions = function() {
      var si = new Crea();
      si.numOrigenes = parseFloat($("input#numVar").val());
      si.numDestinos = parseFloat($("input#numRes").val());


      Crea.current = si;
      $("div#startupSection").hide();
      var inputSection = $("div#inputSection");
      var iSE = inputSection.get(0); // Obtener sin jQuery

      iSE.appendChild(UI.createTextLabel("Costos de transporte"));
      iSE.appendChild(UI.createTable(si.numOrigenes,    si.numDestinos,
"origen")); // Se suma 1 por la fila/columna de totales
      iSE.appendChild(document.createElement("br"));


      var boton = UI.createButton("Resolver", "resolver");
      $(boton).click(function() {
            Crea.tabla = UI.setData();
            Crea.current.run();
            $("#inputSection").hide();
            var resultSection = $("#resultSection");
            console.log(resultSection);
            Crea.current.html += "<p><strong>Terminado</strong></p>";
            resultSection.html(Crea.current.html);
            $("#resultSection").show();
      });
      iSE.appendChild(boton);

      inputSection.show();
```

```
};

UI.createTextField = function(id) {
      var fieldElement = document.createElement("input");
      var field = $(fieldElement);
      field.attr("type", "text");
      field.attr("size", "3");
      if (id) field.attr("id", id);
      return fieldElement;
};

UI.createButton = function(text, id) {
      var fieldElement = document.createElement("input");
      var field = $(fieldElement);
      field.attr("type", "button");
      field.attr("value", text);
      if (id) field.attr("id", id);
      return fieldElement;
};

UI.createTextLabel = function(text) {
      var element = document.createElement("p");
      var p = $(element);
      p.html(text);
      return element;
};

UI.createRow = function(cols, id, index, header) {
      // TODO: Estaba agregando el header de la tabla para introducir
los costos de Crea
      console.log("numCols:" + cols);
      var df = document.createElement("tr");
      $(df).attr("id", id);

      for (var i = 0, len = cols + 2; i < len; i ++) {
            var td = document.createElement(header?"th":"td");
            var temp = UI.createTextField(id + "_var" + i);
            if (header === 1 && i == 0) {
                  $(temp).attr("readonly", "readonly");
                  $(temp).val("De/a");
            } // se salta la esquina
            else if (header === 1 && i === (len -1)) {
                  $(temp).attr("readonly", "readonly");
                  $(temp).val("Oferta");
            }
```

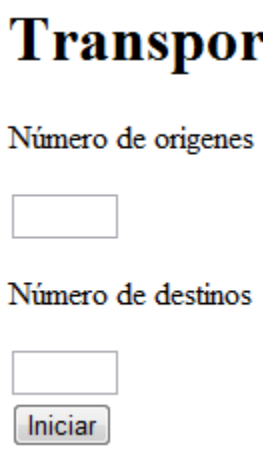
```
            else if (header === 1)  {
                  $(temp).attr("readonly", "readonly");
                  $(temp).val("Destino" + i);
            }
            else if (header === 2 && i == 0)  {
                  $(temp).attr("readonly", "readonly");
                  $(temp).val("Demanda");
            }
            else if (header === 2 && i === (len -1));
            else if (i == 0)  {
                  $(temp).attr("readonly", "readonly");
                  $(temp).val("Origen" + index);
            }
            td.appendChild(temp);
            df.appendChild(td);
      }
      return df;
};

UI.createTable = function(rows, cols, id) {
      var table = document.createElement("table");
      $(table).attr("id", id);

      table.appendChild(UI.createRow(cols, id + "_header", -1, 1));
      for (var i = 0; i < rows; i++) {
            table.appendChild(UI.createRow(cols, id + "_r" + i, i+1));
      }
      table.appendChild(UI.createRow(cols, id + "_totals", -1, 2));
      return table;
};

// ------------ Ejecución, Startup ---------------
//Startup
$(new function() {
      $("div#inputSection").hide();
      $("div#resultSection").hide();
      $('#iniciar').bind("click", UI.showOptions);
});
```

## Manual de Usuario:

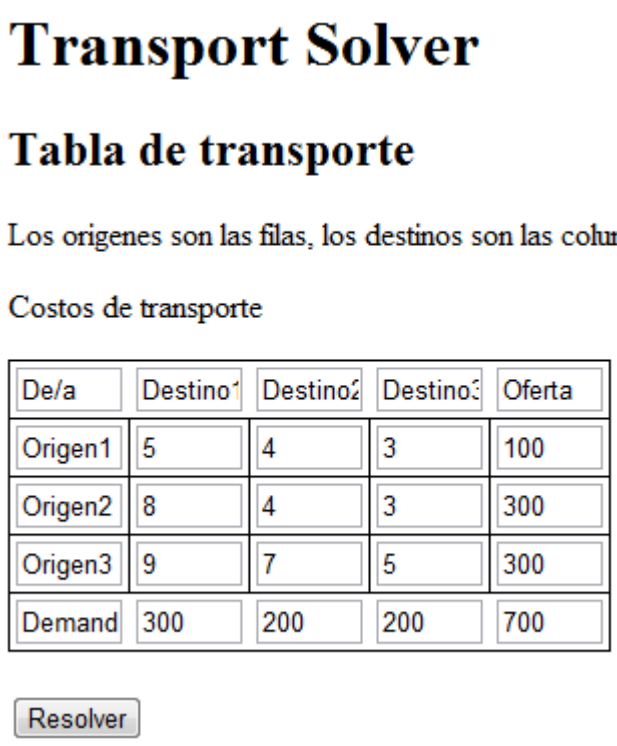
Decidimos hacer nuestro programa para Web por lo que nuestra entrega consta de dos archivos transport.js e index.html, siendo este último el principal.

## Instrucciones de uso:

Al abrir el archivo principal con un explorador, nos presenta la primera pantalla en la que se introduce el número de orígenes y destinos a utilizar en la resolución del problema. Y posteriormente se presiona el botón iniciar.




Esto nos lleva a la forma en la que se llenan la tabla de costos para las rutas. Este programa supone que la tabla se introduce de manera equilibrada.




Al presionar el botón de Resolver nos aparecerán las tablas necesarias para encontrar la solución inicial, y los pasos para llegar a la solución óptima

# Transport Solver

| De/a | Destino1 | Destino2 | Destino3 | Oferta |
|---|---|---|---|---|
| Origen1 | 5 | 4 | 3 | 100 |
| Origen2 | 8 | 4 | 3 | 300 |
| Origen3 | 9 | 7 | 5 | 300 |
| Demanda | 300 | 200 | 200 | 700 |

### Primera Solucion Factible (Northwest Corner)

| De/a | Destino1 | Destino2 | Destino3 | Oferta |
|---|---|---|---|---|
| Origen1 | 5<br>100 | 4 | 3 | 100 |
| Origen2 | 8<br>200 | 4<br>100 | 3 | 300 |
| Origen3 | 9 | 7<br>100 | 5<br>200 | 300 |
| Demanda | 300 | 200 | 200 | 700 |

500+1600+400+700=1000=4200

| De/a | Destino1 | Destino2 | Destino3 | Oferta |
|---|---|---|---|---|
| Origen1 | [5] 100 | [4] | [3] | 100 |
| Origen2 | [8] 100 | [4] 200 | [3] | 300 |
| Origen3 | [9] 100 | [7] | [5] 200 | 300 |
| Demanda | 300 | 200 | 200 | 700 |

500+800+800+900=1000=4000

| De/a | Destino1 | Destino2 | Destino3 | Oferta |
|---|---|---|---|---|
| Origen1 | [5] 100 | [4] | [3] | 100 |
| Origen2 | [8] | [4] 200 | [3] 100 | 300 |
| Origen3 | [9] 200 | [7] | [5] 100 | 300 |
| Demanda | 300 | 200 | 200 | 700 |

500+800+300+1800=500=3900

**Terminado**

Para resolver un nuevo problema solo hay que recargar la página, para regresar a la pantalla de orígenes y destinos.

## Comentarios y Conclusiones:

Para este problema, el programar el algoritmo de cruce del arroyo no fue sencillo. Encontrar los ciclos correctos fue complicado. Ahora sí tomamos más tiempo para realizar el programa, lo cual nos permitió refinar nuestro algoritmo bastante. El principal reto fue programar los métodos para que funcionaran en conjunto.