S I S T E R R

SISTEMA INTEGRATO DI GESTIONE DEI RIFIUTI SOLIDI E RISANAMENTO DEI SITI CONTAMINATI

Sottotema di Ricerca del Progetto VALAMB

Linea 1.3.1.3. - A) Fase 1.3.1.3.2. Individuazione dei modelli

Sviluppo di un prototipo di procedura automatica di calcolo per la pianificazione e la gestione delle attività di raccolta e trasporto dei rifiuti solidi urbani

INDIVIDUAZIONE DEI MODELLI

RELAZIONE RIEPILOGATIVA

Fase 1.3.1.3.2.

Anova

KNOWLEDGE
BASED
SOFTWARE
SOLUTIONS

ANOVA sas di Giovanni Mappa
Centro Direzionale . isola E/5 . scala A . 80143 Napoli
telefono 081 / 7782 111 . 081 / 7782 228
fax 081 / 7782 305 . 081 / 562 72 20
web site www.anova.it
e-mail anova@anova.it
cap. soc. 50.000.000
p.IVA 07385130633
iscritta al n. 611016 REA della C.C.I.A.A.

Data: _____

Referente ANOVA: _____
(dott. Paolo Sabatino)

INDICE

1. IL QUADRO DI RIFERIMENTO PROGETTUALE

1.1 - Riferimenti Generali

Il PARCO SCIENTIFICO E TECNOLOGICO DI SALERNO E DELLE AREE INTERNE DELLA CAMPANIA (denominato PST-Salerno) ha promosso e conseguito l'affidamento, da parte del Ministero dell'Università e della Ricerca Scientifica e Tecnologica (MURST), della realizzazione del Progetto di ricerca "Valorizzazione della qualità delle risorse ambientali e produttive del sistema economico locale e riqualificazione delle peculiari interazioni" - VALAMB.
Questo progetto si articola in due sottotemi di ricerca denominati "Sistema Integrato di Monitoraggio e Gestione dell'Ambiente e del Territorio" - SISTERR e "Impiego dell'osmosi inversa per la concentrazione di succhi di frutta ed ortaggi" - OSMINV.

Per l'esecuzione della ricerca sul sottotema SISTERR, e precisamente per la linea 1.3.1.3. denominata "Sistema di monitoraggio dei bacini idrografici" e le relative fasi 1.3.1.3.2. e 1.3.1.3.3., il PST ha inteso avvalersi della collaborazione di ANOVA sas per la fornitura di servizi di consulenza esperta, analisi, sviluppo e affiancamento per le attività relative alle predette fasi.

Pertanto, In data 8 febbraio 2000, il PST-Salerno ed ANOVA s.a.s. hanno sottoscritto la CONVEZIONE per l'espletamento delle suddette prestazioni.

Documentazione di Riferimento:

a) CONVENZIONE PST-Salerno ANOVA S.a.s.
b) ALLEGATI Tecnici A-B

Gruppo di Lavoro:

Per ANOVA:
dott. Paolo Sabatino (Referente)
ing. Domenico Cianniello

Per PST-Salerno:
ing. Marcello Malangone

1.2 - Obiettivi e Problematiche

La presente RELAZIONE RIEPILOGATIVA si riferisce all'espletamento delle attività per lo ***sviluppo di un prototipo di procedura automatica di calcolo per la pianificazione e la gestione delle attività di raccolta e trasporto dei rifiuti solidi urbani*** (Fase 1.3.1.3.2.), funzionale all'attività di "*Confronto fra i modelli per l'analisi di rete e scelta dei vincoli e del modello da adottare per la raccolta differenziata dei rifiuti*".

Il problema affrontato riguarda, in particolare, lo *sviluppo di una procedura in grado di risolvere il problema della razionalizzazione del servizio di raccolta dei R.S.U., nell'ambito di una rete viaria cittadina complessa, attraverso la ricerca di percorsi ottimali degli automezzi preposti, tali da consentire la minimizzazione di una funzione obiettivo*.

A tal fine si è progettato un modello di calcolo che, interpretando la rete viaria, comprensiva di tutti i vincoli presenti, è in grado di definire mediante metodologie euristiche percorsi tendenti all'ottimale in relazione ai tempi di percorrenza e/o al chilometraggio percorso.

Lo studio è stato condotto sulle basi della ricerca operativa. Le principali difficoltà risiedono nell'applicare le metodologie teoriche di ricerca ad una realtà complessa come può essere quella di un centro urbano.

Nella realizzazione del modello di calcolo è stata soddisfatta l'esigenza di disporre di una metodologia il più possibile flessibile ed allo stesso tempo sufficientemente rapida da fornire in tempi computazionali accettabili la risposta al problema, gestendo un numero elevato di dati dipendente dalle dimensioni dei ambiti di operatività dei mezzi di raccolta.

Per le città con elevato numero di abitanti (elevata quantità di RSU distribuiti su un'ampia area geografica), la raccolta RSU viene organizzata secondo una "zonizzazione" dell'abitato, secondo criteri legati alla densità di popolazione, al tipo di rifiuto prodotto, ecc.

Ogni zona è suddivisa a sua volta in un certo numero di "moduli" (sottoreti di raccolta) coincidenti con l'area di operatività di un singolo automezzo di differente capacità.
L'estensione dei moduli è legata al numero dei cassonetti presenti che dipende dalla capacità dei mezzi di raccolta, scelto sulla base delle dimensioni delle strade che compongono la rete viaria dell'area da servire.

2. LE ATTIVITÀ SVOLTE

Le attività svolte per l'individuazione dei modelli (Fase 1.3.1.3.2.) sono riportate qui di seguito.

2.1 - Analisi dei Requisiti del Prototipo di Procedura Automatica di Calcolo

2.1.1. - Analisi della documentazione del PST-Salerno

Con riferimento alla documentazione del PST-Salerno, riguardante precedenti esperienze su questo tipo di problematiche (v. estratto dalla *Tesi "Analisi delle Tecniche di Ottimizzazione dei Sistemi di Raccolta Differenziata*" e "*Metodologie Euristiche per la Risoluzione del Problema della Ricerca di un Percorso Ottimale*: applicazione ad una zona della città di Roma" - A.Misiti, G.Molinas, P.Viotti), si sono analizzati gli aspetti peculiari della problematica dell'utilizzazione di metodologie euristiche e deterministiche rispetto applicazioni reali.

Si sono potuti mettere a fuoco così, alcuni aspetti "limitanti" degli approcci metodologici già utilizzati per il tipo di problematiche in esame (capacità dei modelli di adattarsi alle esigenze reali di calcolo, tempi di esecuzione, ecc.), di cui tener debitamente conto nella definizione dei modelli per il presente progetto.

2.1.2 - Analisi dello Stato dell'Arte sulla Modellistica disponibile dalla Ricerca Operativa

Il *problema dell'ottimizzazione dei percorsi*, ovvero della ricerca del più breve collegamento tra diversi punti mediante tecniche euristiche, è stato oggetto di molteplici studi.

Secondo la letteratura del settore due sono i principali tipi di approccio ai problemi riconducibili al caso qui trattato:

1) il primo largamente usato e ormai inserito nei metodi della Ricerca Operativa, è quello di schematizzare la rete viaria come un *grafo orientato* G = (N, A), dove con grafo G si intende un insieme N di nodi collegati da un insieme A di archi;
2) il secondo considera i nodi come punti posizionati in uno *spazio cartesiano*, fornendo una sequenza di nodi da visitare senza tenere conto della rete viaria che viene associata in un secondo momento.

La scelta della metodologia è dipendente dalla *funzione obiettivo* che ci si propone di massimizzare o minimizzare, nonché della realtà fisica del problema che è largamente influenzato dal particolare servizio a cui è applicato, sia esso di raccolta che di consegna.

La complessità dello studio, inoltre, cresce notevolmente all'aumentare dei "clienti" da visitare, ed è ulteriormente aggravata se risulta elevata la densità di distribuzione dei clienti sul territorio in esame. In quest'ultimo caso, in particolare *la rete viaria*, con i *vincoli* che presenta, può diventare il parametro fondamentale al quale il sistema deve rapportarsi.

Le prime due fondamentali metodologie di approccio al problema, utilizzate in molteplici applicazioni reali sono riportate qui di seguito.

Il Vehicle Routing Problem (V.R.P.)

Il V.R.P. è il nome dato ad una *metodologia per la trattazione di un'intera serie di problemi che coinvolgono un servizio di distribuzione con una certa periodicità*.

In esso un insieme di clienti dislocato su una vasta area geografica è servito da un gruppo di veicoli che partono da un punto fisso in modo tale da rendere minimi alcuni obiettivi ed allo stesso tempo massimizzare alcune priorità, ad esempio:

a. *massimizzare la somma delle priorità dei clienti che possono essere visitati dall'insieme dei veicoli*
b. *minimizzare i costi fissi relativi ai veicoli*
c. *minimizzare i costi variabili dei tragitti.*

Possono anche essere considerati obiettivi composti quali:

1. *minimizzazione del costo fisso dei veicoli impiegati con la massimizzazione delle priorità (a+b);*
2. *minimizzazione dei percorsi e del tempo, cioè dei costi variabili, con la massimizzazione delle priorità (a+c);*
3. *minimizzazione del costo fisso dei veicoli impiegati insieme alla minimizzazione dei costi variabili e alla massimizzazione delle priorità (a+b+c).*

La minimizzazione dei soli costi (b+c) non viene presa in esame perché comporterebbe la soluzione banale 0. Il V.R.P. è quindi, per la sua impostazione, adatto in genere per definire le "zone di operatività" dei diversi mezzi necessari per l'espletamento del servizio, ed è una metodologia impiegata generalmente su grandi aree geografiche ove è possibile trascurare i vincoli imposti dalla viabilità.

Il Travelling Salesman Problem (T.S.P)

Definito un grafo non orientato (definito come un grafo in cui l'arco che collega due nodi N_i e N_j può essere percorso da N_i a N_j e viceversa) G = (N, A), con N l'insieme dei nodi e con A l'insieme degli archi, il *problema del Commesso Viaggiatore* è quello di determinare il ciclo hamiltoniano di tempo, costo o lunghezza minima, dove per ciclo hamiltoniano si intende un ciclo che visita tutti gli *n* nodi del grafo una e una sola volta.
La generalizzazione del T.S.P., conosciuta come G.T.S.P., si ottiene quando, oltre al tempo di percorrenza sugli archi del grafo, si ha anche un "punteggio" associato ad ogni nodo ed un tempo massimo di percorrenza del ciclo (o cammino). In tal modo, poiché non risulta sempre possibile visitare tutti i nodi, la soluzione a cui si arriva è l'individuazione di un percorso con il massimo "punteggio". Se nell'ambito di questo percorso si prende in considerazione anche la finalità tempo, si giunge all'individuazione di un cammino che massimizza la differenza tra il punteggio associato ai nodi visitati nel ciclo e quello associato al tempo di percorrenza.

Metodologie Euristiche e Algoritmi di costruzione dei percorsi

Per la soluzione dei *problemi applicativi* generalmente ci si avvale di *procedure euristiche* che forniscono soluzioni approssimate.
Le euristiche presenti in letteratura possono essere raggruppate in tre classi:

a. *euristiche di costruzione del cammino: mediante un criterio di scelta vengono inseriti nuovi nodi nel percorso, verificando sempre che sia mantenuta l'ammissibilità;*
b. *euristiche di miglioramento: dato un cammino ammissibile, cercano di determinarne uno migliore*
c. *euristiche miste che implicano entrambe le precedenti metodologie.*

La complessità di tali euristiche può essere considerata di ordine variabile tra $O(n^2)$ e $O(n^3 \log n)$.
La procedura di risoluzione del problema può essere considerata divisa in due parti sequenziali:

1) *la prima basata su criteri di scelta che definisce il nodo o i nodi da inserire nel percorso;*

2) *la seconda è la costruzione vera e propria della sequenza di nodi che costituiscono il tragitto.*

La metodologia per la costruzione mediante queste procedure di una soluzione tendente all'ottimale potrà essere quindi basata, come propone la letteratura del settore, sui seguenti criteri:

a) risparmi

b) extrachilometraggio

c) extratempo

♦ Il ***Criterio del Risparmio*** è stato trattato da ***Clarke-Wright*** e si basa su semplici considerazioni.

Sia E un insieme di clienti x_i con i=1,…,n. Supponiamo inizialmente di impiegare n veicoli, ognuno dei quali serva un solo cliente. In queste condizioni, se x_0 è la posizione del deposito da cui partono i mezzi e d(x_0, x_i) è la distanza tra il deposito ed il cliente x_i, la lunghezza totale del percorso sarà quindi:

$$L_{tot} = 2\sum_{i=1}^{n} d(x_0, x_i)$$

Se si ipotizza ora che un mezzo può servire solo 2 clienti: nel caso di un viaggio solo per ogni cliente si avrà, nel caso di un unico viaggio:

$$d_{tot} = 2 \cdot d(x_0, x_i) + 2 \cdot d(x_0, x_j)$$

$$d'_{tot} = d(x_0, x_i) + d(x_i, x_j) + d(x_j, x_0)$$

avremo quindi, che il risparmio è:

$$R(x_i, x_j) = d_{tot} - d'_{tot} = d(x_0, x_i) + d(x_j, x_0) - d(x_i, x_j)$$

i clienti da servire x_i e x_j saranno quindi scelti sulla base del massimo valore di $R(x_i, x_j)$.

In realtà, il risparmio così ottenuto non è corretto quando il grafo orientato non risulta completamente connesso, perché in generale $d(x_i, x_j) \neq d(x_j, x_i)\ \forall i,j$. Pertanto, la d_{tot} come calcolata da Clarke-Wright non è più valida, e deve essere sostituita da:

$$d_{tot} = \left[d(x_0, x_i) + d(x_i, x_0)\right] + \left[d(x_0, x_j) + d(x_j, x_0)\right]$$

quindi, il risparmio diventa:

$$\boxed{R(x_i, x_j) = d(x_i, x_0) + d(x_0, x_j) - d(x_i, x_j)}$$

da cui in generale $R(x_i, x_j) \neq R(x_j, x_i)$.

♦ Diversa è l'impostazione data dal ***Criterio dell'Extrachilometraggio***.

Consideriamo anche adesso un deposito x_0 e un insieme di clienti x_i con i=1,…,n. Nell'ipotesi in cui ogni veicolo servisse due clienti consecutivi, anche se inseriti già in un percorso, si consideri la maggiorazione di percorso che comporterebbe il servizio ad un terzo cliente.

Si può ottenere l'algoritmo semplicemente calcolando la differenza tra il percorso base e quello aggiornato.

Il percorso base iniziale, considerando solo il servizio ai due clienti x_i e x_j:

$$l = 2 \cdot d(x_i, x_j)$$

il percorso aggiornato considerando l'inserimento di x_k:

$$l' = d(x_i, x_j) + d(x_j, x_k) + d(x_k, x_i)$$

l'extrachilometraggio, perciò, risulta:

$$ec(x_i, x_j, x_k) = l' - l = d(x_j, x_k) + d(x_k, x_i) - d(x_i, x_j)$$

Questa definizione che troviamo in letteratura ha lo stesso problema della definizione del risparmio, e deve essere corretta se abbiamo un grafo orientato non completamente connesso. Infatti, risulta:

$$l = d(x_i, x_j) + d(x_j, x_i)$$

pertanto, l'extrachilometraggio diventa:

$$\boxed{ec(x_i, x_j, x_k) = d(x_j, x_k) + d(x_k, x_i) - d(x_j, x_i)}$$

♦ Il ***Criterio dell'Extratempo*** risulta essere una estensione del criterio precedente ed è funzione della variabile tempo. Infatti alle distanze vengono associate le velocità di percorrenza corrispondenti alla fascia oraria in cui il mezzo transita nell'arco, e vengono determinate le maggiorazioni temporali necessarie a servire il cliente *k* inserito tra i nodi *i* e *j*.
L'extratempo si esprime nella seguente forma:

$$\boxed{et(x_i, x_j, x_k) = d(x_j, x_k)/v_{j,k} + d(x_k, x_i)/v_{k,i} - d(x_j, x_i)/v_{j,i}}$$

La costruzione del percorso, una volta individuato il nodo o la sequenza di nodi da inserire avviene con procedure che possono essere di due tipi:

a. *procedure parallele;*
b. *procedure sequenziali.*

Le procedure parallele costruiscono più tragitti contemporaneamente fino ad ottenere il percorso finale. Queste possono essere divise principalmente in due categorie:

1. *viene fissato a priori il numero k di tragitti, con k≤n, dove n è il numero dei nodi; i tragitti crescono simultaneamente per l'immissione ad ogni fase di un altro cliente fino a che tutti i clienti sono inseriti in un tragitto;*
2. *vengono costruiti percorsi brevi tra il nodo ingresso ed ogni altro cliente, e poi vengono uniti per formarne uno più grande; in questo modo non è possibile stabilire a priori il numero di tragitti risultante.*

Le procedure sequenziali costruiscono il tragitto inserendo i nodi in forma sequenziale, secondo la funzione obiettivo e con l'ausilio di uno dei criteri riportati, fino a che tutti i clienti non sono inseriti nel tragitto.

2.2 - Progettazione e Sviluppo delle Procedure di Input

In generale, i dati indispensabili per lo sviluppo di opportuni algoritmi per la risoluzione del problema posto, derivano dalla:

- *Caratterizzazione delle utenze*
- *Caratterizzazione della rete*
- *Caratterizzazione della raccolta.*

2.2.1 - Caratterizzazione delle utenze

La caratterizzazione delle utenze ha come principali parametri di riferimento la popolazione servita e le caratteristiche di produzione dei rifiuti. In particolare è possibile schematizzare tali informazioni in relazione a:

- *Numero di abitanti e tipologia di utenza suddivisi per zone aventi densità demografica omogenea; utenze domestiche (centro storico, zone residenziali), utenza non domestica (esercizi commerciali, ristoranti, uffici), fluttuazioni stagionali;*
- *Stima e composizione qualitativa dei rifiuti prodotti per ogni singola utenza e strutturati per ogni sottoarea individuata.*

2.2.2. - Caratterizzazione della raccolta

La caratterizzazione della raccolta richiede la definizione completa della tipologia di raccolta a cui gli algoritmi sono applicati, le caratteristiche dei mezzi di raccolta, la frequenza della raccolta, la presenza di eventuali stazioni di compattazione.

In particolare, si ritiene di rendere il sistema in progetto applicabile alle seguenti metodologie di raccolta:

- *Sistema a tre cassonetti: indifferenziato, F.O.R.S.U., R.D.M., integrato con campane per la raccolta del vetro e/o della carta;*
- *Sistema a due cassonetti: indifferenziato, F.O.R.S.U., integrato con campane per la raccolta del vetro e/o della carta;*
- *Sistema tradizionale con la raccolta indistinta di tutti i R.S.U..*

I sistemi di raccolta elencati possono inoltre essere previsti mediante servizio stradale, con posizionamento dei contenitori lungo l'asse stradale o nel caso di servizio a grandi utenze presso le singole utenze.
La flessibilità del sistema previsto è inoltre tale da prevederne un possibile utilizzo anche in relazione alle problematiche di compattazione e trasporto.

2.2.3 - Caratterizzazione della rete

La caratterizzazione della rete richiede la definizione delle caratteristiche intrinseche quali le *caratteristiche delle strade*, le *caratteristiche dei cassonetti* e di quelle correlate a fattori esterni come la caratterizzazione della *velocità di percorrenza*, della *potenzialità dei mezzi di raccolta* e dei *costi di percorso per unità*.

La rappresentazione completa del *grafo stradale* richiede una strutturazione in relazione a nodi ed archi a cui vanno collegate una serie di dati ed informazioni relative alle caratteristiche geometriche, di viabilità e di velocità media di percorrenza.

Lo sviluppo degli algoritmi non può prescindere dalla definizione di alcune scelte metodologiche preliminari:

- *suddivisione del territorio urbano in aree contraddistinte per omogeneità su quantità e tipologia di rifiuti prodotti e caratteristiche del tessuto urbano;*
- *scelta del tipo di raccolta da adottare per ogni sottoarea individuata (domiciliare, stradale, monomateriale, multimateriale (due o tre cassonetti));*
- *numero di contenitori che è necessario posizionare al fine di garantire il servizio per ogni utenza o gruppo individuato;*

- *posizionamento dei cassonetti e individuazione dei bacini di utenza e dei carichi gravanti su di essi;*
- *frequenza minima della raccolta per ogni tipologia di rifiuto e di sistema operativo di raccolta adottato.*

Il problema della considerevole quantità di dati da caricare per la caratterizzazione topologica e geometrica della rete (e in visione di una implementazione SIT la possibilità di completamento con dati georeferenziati) è risolto con un sistema di caricamento dati di tipo CAD, su cui ottenere la possibilità di una verifica di congruenza dei dati caricati.

I dati che caratterizzano il sistema di raccolta (utenza, rete, mezzi) sono organizzate in tabelle relazionate, nelle quali vengono riportati: dati che servono specificamente per la fase di input dell'algoritmo e dati che servono esclusivamente per la rappresentazione grafica del sistema o che contengono altri tipi di informazione.

Dati relativi all'utenza

Utenza	*Parametri principali di riferimento*	*Altro*
Domestica	densità popolazione residente incidenza su arco incidenza su nodo	Tipologia di rifiuto associata, parametro di caratterizzazione quantitativo, Possibilità di tarare il sistema sulla base dei dati storici e di esercizio riscontrabili
Esercizi commerciali	Superficie	Tipologia di rifiuto associata
Esercizi di ristorazione	numero di coperti	Tipologia di rifiuto associata
Uffici	numero di impiegati	Tipologia di rifiuto associata

Per quanto riguarda i parametri che caratterizzano il tipo di utenza e il tipo di rifiuto ad essa associato non è prevista una procedura che fornisca tali dati come input bensì è possibile collegare ad ogni cassonetto (ID cassonetto) il tipo di utenza servita (identificandola ognuna con un colore diverso) e indicare, ad esempio, il numero di utenti serviti (nel caso di utenza domestica) con un cerchio il cui diametro è proporzionale agli utenti stessi.
In questo modo è possibile attraverso una rappresentazione grafica vedere per un determinato cassonetto quale area o zona della rete copre e verificare, sempre graficamente, se è stata effettuata una corretta distribuzione dei cassonetti sulla rete.

Le informazioni che riguardano la tipologia e la quantità di rifiuti prodotti possono essere organizzate in tabelle, divise per tipo di utenza, dove vengono registrati i dati che si ritengono più affidabili relativamente alla realtà territoriale in esame.
Nella fase iniziale è possibile caricare dati bibliografici per le utenze domestiche e dati derivanti dal MUD per utenze non domestiche, salvo aggiornamento derivante da specifiche analisi di campo (previste nella fase successiva).

Relazioni Cassonetto-Utenza

ID CASSONETTO	TIPO DI UTENZA	Parametro riferimento	Colore identificativo	AREA INFLUENZA
001	Domestica1 (centro storico)	Densità abitativa [n ab/m^2]	Rosso	πR^2

$$r = \sqrt{\frac{c \cdot V_{cass.} \cdot p}{3.14 \cdot g \cdot \left(\frac{1}{f}\right) \cdot d}}$$

dove:

r: raggio dell'area da servire (supposta circolare) [m]
V_{cass}: volume cassonetto [m^3]
p: peso specifico rifiuto [kg/m^3]
g: produzione di rifiuto giornaliera procapite
c: coeff. di utilizzazione del cassonetto
f: frequenza di raccolta [operazioni/giorno]
d: densità abitativa della zona [ab/m^2].

Dati relativi alla raccolta e alla rete

NODI: i nodi possono indicare la posizione di un cassonetto, un incrocio stradale, un punto di ingresso alla rete (inizio percorso di raccolta), un punto finale della rete (impianto di trasformazione, ecc.). Per il nodo incrocio, ovviamente, il volume è pari a zero e il tempo di svuotamento corrisponde al tempo di attraversamento; per il nodo finale il tempo può indicare i minuti necessari per lo scarico dei rifiuti ed eventualmente il tempo per tornare sul circuito di raccolta (nell'ultima colonna è specificato quale delle informazioni va fornita in input all'algoritmo di risoluzione dei percorsi ottimi):

		INPUT ALGORITMO
ID nodo [num]	001	SI
Tipo di nodo [stringa]	cassonetto	SI
Tipo raccolta [stringa]	R.D.M.	NO
Tipo o numero utenti serviti [num]	200 abitanti	NO
X [num]	32 m	NO
Y[num]	100 m	NO
Volume netto utilizzabile [num]	25 mc	SI
Frequenza di rimozione [num]	2 volte/settimana	NO
Superficie servita [num]	500 mq	NO
Tempo di sosta [num]	1.5 min.	SI
Nome strada [stringa]	Via Roma	NO

Queste informazioni sono organizzate in una tabella dei nodi che, come si nota, è composta da informazioni che servono per la sola rappresentazione grafica ed altre informazioni necessarie come input dell'algoritmo.
Altre informazioni supplementari possono riguardare lo stato del cassonetto, la data d'acquisto, le caratteristiche di svuotamento, ecc.

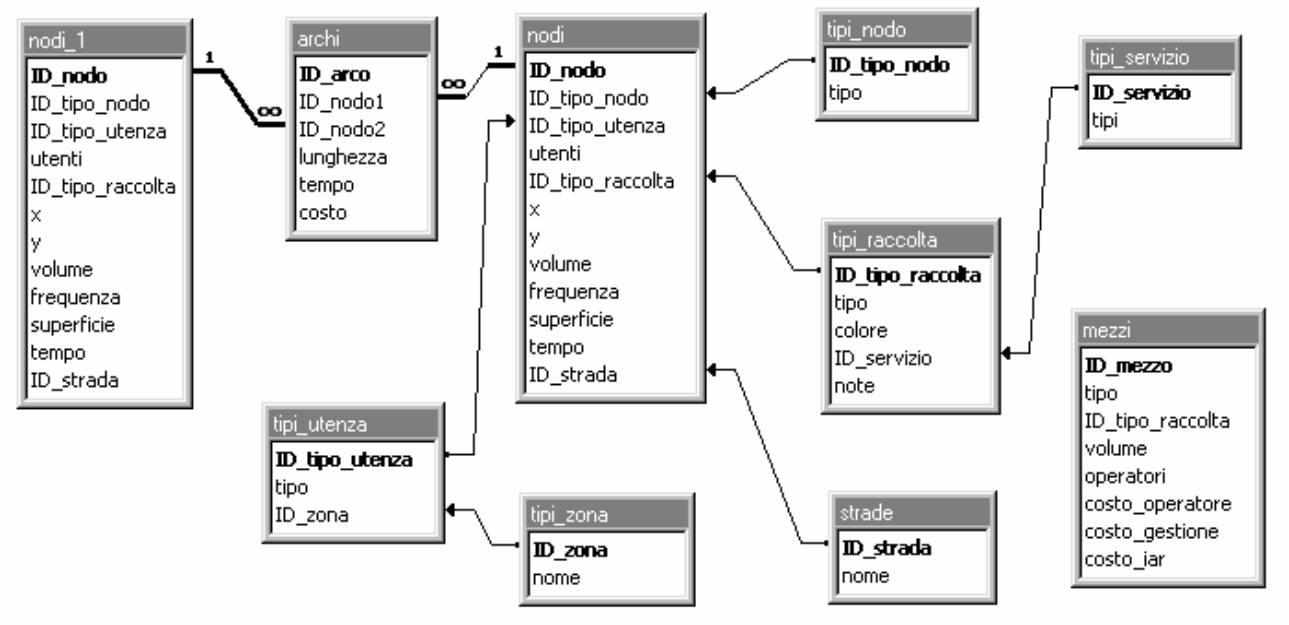
ARCHI: individuati da caratteristiche topologiche e funzionali:

		INPUT ALGORITMO
ID arco [num]	001	SI
Identificazione [stringa]	Via Roma	NO
Nodo iniziale [ID nodo]	001	SI
Nodo finale [ID nodo]	002	SI
Lunghezza [num]	650	SI
Costo di percorrenza [num]	1000 Lit/Km	SI
Tempo di percorrenza [num]	3 minuti	SI

MEZZI: i mezzi sono caratterizzati da proprietà tecniche (volume, tipo di raccolta) e da proprietà economiche (costi fissi); ad esempio:

			INPUT ALGORITMO
ID mezzo [num]	01	Caratteristiche tecniche	SI
Capacità utile [num (m3)]	20		SI
Tipo raccolta [stringa]	R.D.M.		NO
Numero operatori [num]	2		NO
Costo d'investimento, ammort. e reint. [num]	150	Caratteristiche economiche (costi fissi)	NO
Costo operatori [num]	50		NO
Costo di gestione [num]	1		NO

Il diagramma delle relazioni tra queste e le altre tabelle del database:



2.3 - Progettazione e Sviluppo delle Procedure di Output

Gli output del sistema dovranno consentire la *scelta ottimale delle attività di dimensionamento e di gestione della raccolta e trasporto* su di un territorio a scala provinciale.

Il dimensionamento comporterà la definizione delle infrastrutture minime necessarie ed il confronto fra opzioni diverse (tipologia di raccolta, mezzi, numero dei cassonetti, collocazione, ecc.) sulla base delle procedure di ottimizzazione dei costi.

Il controllo sulle attività di gestione invece permetterà l'individuazione dei minimi costi di gestione sulla base delle informazioni di input, delle infrastrutture disponibili, della topologia della rete, ecc. e consentirà il confronto dei percorsi e dei risultati in funzione di scelte diverse, dell'interposizione di stazioni di compattazione, ecc.

Gli output numerici e grafici consentiranno una immediata leggibilità dei percorsi.

Gli output grafici saranno costituiti dalla visualizzazione diretta sulla mappa stradale del percorso o dei percorsi ottenuti, evidenziati con colori opportuni.

Gli output numerici saranno organizzati in finestre che riportano: caratteristiche dei mezzi scelti, tempo presunto per la raccolta, volume totale di rifiuti raccolti organizzati per tipologia, numero di cassonetti visitati, lunghezza totale del percorso, costo presunto per ogni circuito.
Un ulteriore output numerico riporterà in sequenza i cassonetti da visitare, con informazioni riguardanti il nome della strada dove è situato, il volume ed eventualmente distanza progressiva e parziale del percorso da effettuare.

2.4 - Ingegnerizzazione e Testing del Prototipo di Procedura Automatica di Calcolo

2.4.1 - Implementazione del Modello di Calcolo (v. Codice in Allegato)

Il problema da affrontare è, come già enunciato, la *razionalizzazione del servizio di raccolta dei R.S.U. mediante la ricerca di percorsi ottimali per gli automezzi preposti, tali da consentire la minimizzazione di una funzione obiettivo*.
A tal fine si è progettato un *modello di calcolo* che, interpretando la rete viaria, comprensiva di tutti i vincoli presenti, è in grado di definire mediante metodologie euristiche percorsi tendenti all'ottimale in relazione ai tempi di percorrenza e/o al chilometraggio percorso e/o dei costi.

Lo studio è stato condotto sulle basi della ricerca operativa; le principali difficoltà da superare consistono nell'applicare le metodologie teoriche di ricerca ad una realtà complessa come può essere quella di un centro urbano.

Schematicamente il problema che si presenta allo studio è il seguente: *data un'area geografica dove sono situati n nodi (clienti o cassonetti) collegati tra loro da una rete viaria complessa, trovare il minimo percorso del mezzo di raccolta necessario a visitare almeno una volta ciascun nodo*.

Nella realizzazione del modello di calcolo è stata soddisfatta l'esigenza di disporre di una metodologia il più possibile flessibile ed allo stesso tempo sufficientemente rapida, tali da fornire in tempi computazionali accettabili, la risposta al problema.
Il modello deve gestire un numero elevato di dati dipendente dalle dimensioni dei moduli o della zona di raccolta, a seconda se si vuole trovare il percorso ottimale sul modulo singolo predefinito, o se si desidera ricercare i percorsi che permettano la raccolta su un'intera zona, che non sia stata suddivisa in moduli.

Il linguaggio utilizzato per l'implementazione dell'algoritmo è *Visual Basic* per la possibilità di combinare la costruzione visiva dell'applicazione, disegnando direttamente le finestre dei dati necessari alla rappresentazione e i loro componenti.

I dati relativi ai parametri necessari per la definizione del sistema sono organizzati in tabelle richiamabili attraverso delle finestre di tipo user-friendly (ad esempio finestra degli utenti, dei nodi, dei cassonetti, dei mezzi) con la possibilità, per semplificare le procedure di input da parte dell'utente di selezionare il numero ed il tipo di veicoli che intende utilizzare, tra quelli disponibili, visualizzati in una apposita maschera di interfaccia che riporta le informazioni relative ai mezzi selezionati. Inoltre è possibile per l'utente aggiornare o modificare i dati contenuti nelle finestre.

Una possibilità di organizzazione dei dati è riportata nella seguente tabella:

RACCOLTA R.D.M. **MEZZI**					
Mezzo	*Capacità [mc]*	*Numero Operatori*	*Costo [Lit*10^6]*		*Mezzi scelti per la raccolta*
01	14	2	100	→	Mezzo 2 - 20 mc
02	20	3	200		
03	…	…	…		

I tabulati (reports) che il programma è in grado di generare oltre alla sequenza dei nodi visitati, riporta per ciascun nodo:

a. i percorsi trovati per ogni camion
b. il tempo di percorrenza
c. i nodi da caricare
d. la quantità totale dei rifiuti raccolti

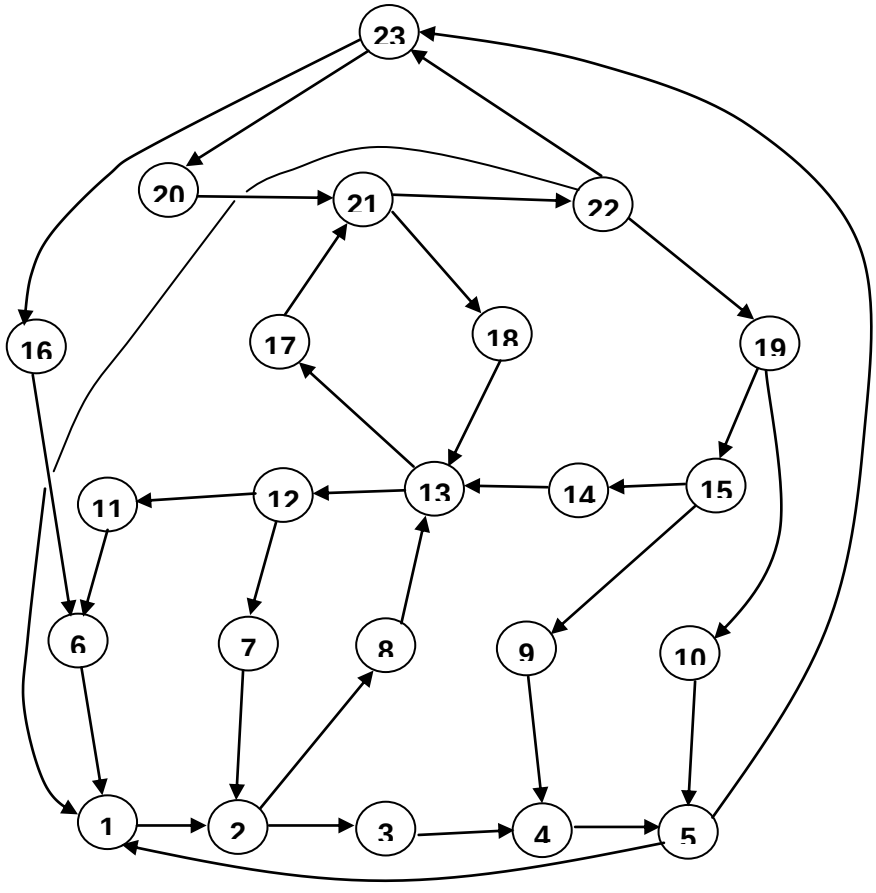
La versione definitiva riporterà anche:

a) *la localizzazione del nodo;*
b) *l'orario di partenza dal nodo per la continuazione del tragitto;*
c) *il numero dei cassonetti svuotati dall'inizio della raccolta al momento della partenza dal nodo;*
d) *la quantità dei rifiuti raccolta in mc e in %;*
e) *la capacità residua del mezzo.*

Si sottolinea che, dove un qualsiasi nodo, in cui siano localizzati uno o più cassonetti, venga visitato più di una volta, lo svuotamento dei cassonetti viene effettuato al primo passaggio tranne in presenza di particolari vincoli stabiliti a priori o decisi dall'Azienda incaricata della raccolta, ed in questa fase vengono aggiornati i valori descritti nei cinque punti sopra.

2.4.2 - Ingegnerizzazione e Testing del Modello di Calcolo

La rete di calcolo sulla quale è stato effettuato il *testing* degli algoritmi viene derivata da una rete stradale reale (rete urbana di Portici), costituita da 23 nodi cassonetto e 33 archi orientati. La prima rete rispetto alla seconda è semplificata rispetto ai *nodi incrocio*.
In altri termini, si è convertita la rete reale in quella di calcolo attraverso un algoritmo che genera nuovi archi che collegano solo i nodi cassonetto, inglobando in se gli archi di partenza e i nodi incrocio.
Si è utilizzato tale approccio per poter avere una riduzione del numero di nodi complessivi e per snellire le procedure di calcolo inerenti alla ricerca dei percorsi ottimali.
In tale rete il nodo 1 funge sia da nodo ingresso, sia da nodo cassonetto, sia da nodo di uscita. La procedura non sarebbe comunque cambiata se non ci fosse stata coincidenza tra il nodo di ingresso e il nodo di uscita.



3. METODOLOGIE, CRITERI E STRUMENTI ADOTTATI

La definizione dei parametri di input e dei dati necessari alla descrizione numerica del dominio di operatività sono una delle parti più importante del problema.

L'analisi si svolge essenzialmente secondo le seguenti fasi:

1. *Definizione della zona di operatività, funzione delle caratteristiche del mezzo di raccolta, delle disposizioni esistenti all'interno della struttura organizzativa dell'Azienda incaricata della raccolta, delle dimensioni dei contenitori di raccolta dislocati o da dislocare e del loro numero. Importante in questa fase è uno studio merceologico del rifiuto, dei quantitativi di raccolta nell'arco della settimana tra i vari periodi dell'anno, per una ottimale scelta della tipologia dei contenitori e del loro posizionamento.*
2. *Orari di lavoro, numero di addetti, numero di veicoli.*
3. *Dati relativi alle operazioni di caricamento, svuotamento, scaricamento e riposizionamento dei cassonetti.*
4. *Studio della viabilità delle singole zone (sensi di percorrenza, segnaletica, ecc.). Rilevamento della lunghezza delle strade e delle intensità di traffico, quest'ultima necessaria per la definizione delle velocità medie di percorrenza.*
5. *Definizione su appropriata cartografia dei nodi per la codifica della viabilità da fornire all'elaboratore; determinazione dei possibili collegamenti diretti tra i singoli nodi; misura delle relative distanze.*
6. *Inserimento dati*
7. *Riproduzione dei percorsi ottenuti su cartografia.*

3.1 - Definizione dei nodi

La schematizzazione numerica delle zone di operatività è rappresentata da un *grafo orientato non completamente connesso* G=(N,A,D,V) dove N è l'insieme dei nodi, A è l'insieme degli archi dei collegamenti diretti tra i nodi, a cui sono associate le distanze D e le velocità V.

Si è limitato l'impiego di un numero eccessivo di nodi per non moltiplicare in modo eccessivo l'occupazione di risorse e i tempi di calcolo; infatti, come già detto in precedenza, la *complessità del problema è di ordine esponenziale funzione del numero di nodi*.

È stata effettuata, inoltre, per lo stesso motivo, la scelta di aggregare più cassonetti in un unico nodo. I nodi vengono individuati perciò sulla cartografia relativa alla zona di operatività del mezzo; i collegamenti sono determinati in base alla viabilità locale e non si hanno limitazioni sul numero di collegamenti a cui un nodo può essere soggetto, a parte il numero totale dei nodi.

La definizione dei possibili collegamenti tra i nodi è fondamentale ai fini di una corretta risoluzione del problema; da essa dipendono, infatti, le diverse successioni di nodi che contribuiscono alla definizione dei tragitti.

Sono state considerate tutte le infrastrutture viarie utilizzabili (funzione anche della larghezza di transito e delle dimensioni del mezzo) al fine di assicurare il maggior numero possibile di percorsi alternativi.

Non è stata presa in considerazione la possibilità di effettuare manovre a zig-zag (cioè di poter effettuare la raccolta su entrambi i lati con attraversamenti) durante la raccolta nelle strade a senso unico di marcia, e nemmeno di poter trasportare il cassonetto da un lato all'altro della strada per effettuarne lo svuotamento.

Strade a doppio senso sono quindi considerate come divise da una ideale linea di separazione, e nodi presenti su due lati opposti non hanno la possibilità di essere collegati direttamente.

Tutta la viabilità risulta quindi schematizzata mediante una tabella A delle distanze tra nodo 1 e nodo 2.

3.2 - Definizione parametri di input

I parametri che intervengono nel modello sono i seguenti:

a. *velocità di spostamento del mezzo;*
b. *tempo di svuotamento del singolo cassonetto;*
c. *capacità del mezzo utilizzato per la raccolta;*
d. *capacità del singolo contenitore;*
e. *tempo necessario all'arresto e partenza del mezzo di raccolta.*

Per la definizione dei valori corrispondenti ai suddetti punti si intende avvalersi di dati direttamente rilevati sulla zona di operatività. Una considerazione particolare merita la velocità di spostamento del mezzo che è un parametro importantissimo ai fini del tempo di percorrenza.
Essendo la raccolta effettuata durante le ore diurne nella maggioranza delle zone cittadine, sono state previste fasce di velocità medie relative ad ogni arco di collegamento tra due nodi ed associate ad altrettanti intervalli orari.
Queste velocità sono legate in modo inverso alle diverse intensità di traffico presenti nell'arco stradale.
In questa fase è stato possibile anche tenere conto delle differenti operazioni dipendenti dalle realtà locali, come le limitazioni sulla viabilità in determinate fasce orarie (entrate e uscita dalle scuole, dai posti di lavoro, mercati, ecc.).

3.3 - Procedura di calcolo

Si è cercato di soddisfare l'esigenza di disporre di una metodologia il più flessibile possibile ed allo stesso tempo sufficientemente rapida da fornire in tempi di calcolo accettabili la risposta al problema, consentendo allo stesso tempo di gestire un numero elevato di dati.
Non ultima l'esigenza di tenere in conto l'impiego del programma anche da parte di operatori non specializzati.
A questo scopo si è progettato una interfaccia operatore-macchina che permette di gestire completamente sia la fase di immissione dati e variazione degli stessi, sia la realizzazione in forma autonoma di un archivio di informazioni sui singoli moduli, zone e circoscrizioni.

La procedura è suddivisa in due fasi principali:

a. *immissione dati*
b. *costruzione dei percorsi e scelta del percorso che soddisfa la funzione obiettivo.*

Nella prima fase si inseriscono esclusivamente le informazioni base per la schematizzazione della planimetria stradale ed i parametri di input, in particolare:

- *la misura della distanza tra un nodo e gli altri nodi;*
- *la velocità media di percorrenza dei singoli archi associati alle diverse fasce orarie;*
- *il numero dei cassonetti eventualmente presenti nel nodo;*
- *la denominazione del nodo in riferimento alla toponomastica stradale;*
- *la capacità del mezzo;*
- *la capacità del singolo cassonetto;*
- *il tempo medio necessario per l'arresto, lo svuotamento del singolo cassonetto e la partenza del mezzo.*

Dopo l'immissione dati comincia la costruzione dei percorsi dei percorsi parziali (seconda fase b.). Prima di entrare in dettaglio, introduciamo la seguente definizione: dati due nodi

generici *i* e *j*, si definisce "*arco fondamentale* A_{ij}" l'arco che collega direttamente la coppia di nodi N_i e N_j.
L'algoritmo impiegato sfrutta un sistema di risoluzione del problema basato su matrici; partendo dalla matrice formata dagli archi fondamentali indicata di ordine zero C^0, arriva a definire un matrice di ordine r (con $0 < r < n$) C^r, il cui elemento C_{ij} contiene la distanza (o il tempo) per il collegamento dei generici nodi N_i e N_j realizzato con r+1 archi. La matrice potrà raggiungere l'ordine massimo di (n-1) con n il numero di nodi; inoltre, non sarà sempre necessario che si arrivi al massimo ordine potendo fermare il calcolo non appena risulti $C^k=C^{k-1}$.

Per descrivere il metodo occorre introdurre la matrice delle distanze $D=[d_{ij}]$ e l'operazione tra matrici:

$$Z = X \bullet Y$$

dove $X=[x_{ij}]$, $Y=[y_{ij}]$ sono le matrici C^k, e $z_{ij} = \min_{1<k<n}[x_{ik} + y_{kj}]$. Eseguendo l'operazione:

$$C^2 = C \bullet C$$

è chiaro che gli elementi di C^2 rappresentano le distanze minore di ordine 2 (cioè collegati da 3 archi fondamentali) tra ogni coppia di nodi. Allo stesso modo:

$$C^k = C^{k-1} \bullet C$$

rappresenta la matrice delle distanze minime ottenute percorrendo al massimo k archi.

Per il calcolo di C^k con $k \leq n$-1, si è impiegato l'algoritmo di Floyd-Warshall che risolve il problema in forma semplificata.

Per motivi legati all'occupazione di memoria, essendo il programma progettato per PC con sistema operativo Windows, le matrici non verranno allocate direttamente nella memoria RAM del computer, ma su file del disco rigido.
In questo modo l'aumento dei tempi di calcolo è accettabile e si ha un limite molto alto (spazio libero nel disco fisso) sul numero n dei nodi che possono essere trattati, senza esaurire le risorse del sistema.

Nell'ultima parte viene impiegato un criterio di scelta per il nodo che deve essere inserito nel percorso in costruzione mediante una procedura sequenziale.

Si sono considerate come *funzione obiettivo da minimizzare* quelle del minor "tempo", ma ciò non toglie la possibilità che la funzione obiettivo possa essere i "costi" o "lunghezza", per l'effettuazione del percorso di raccolta.

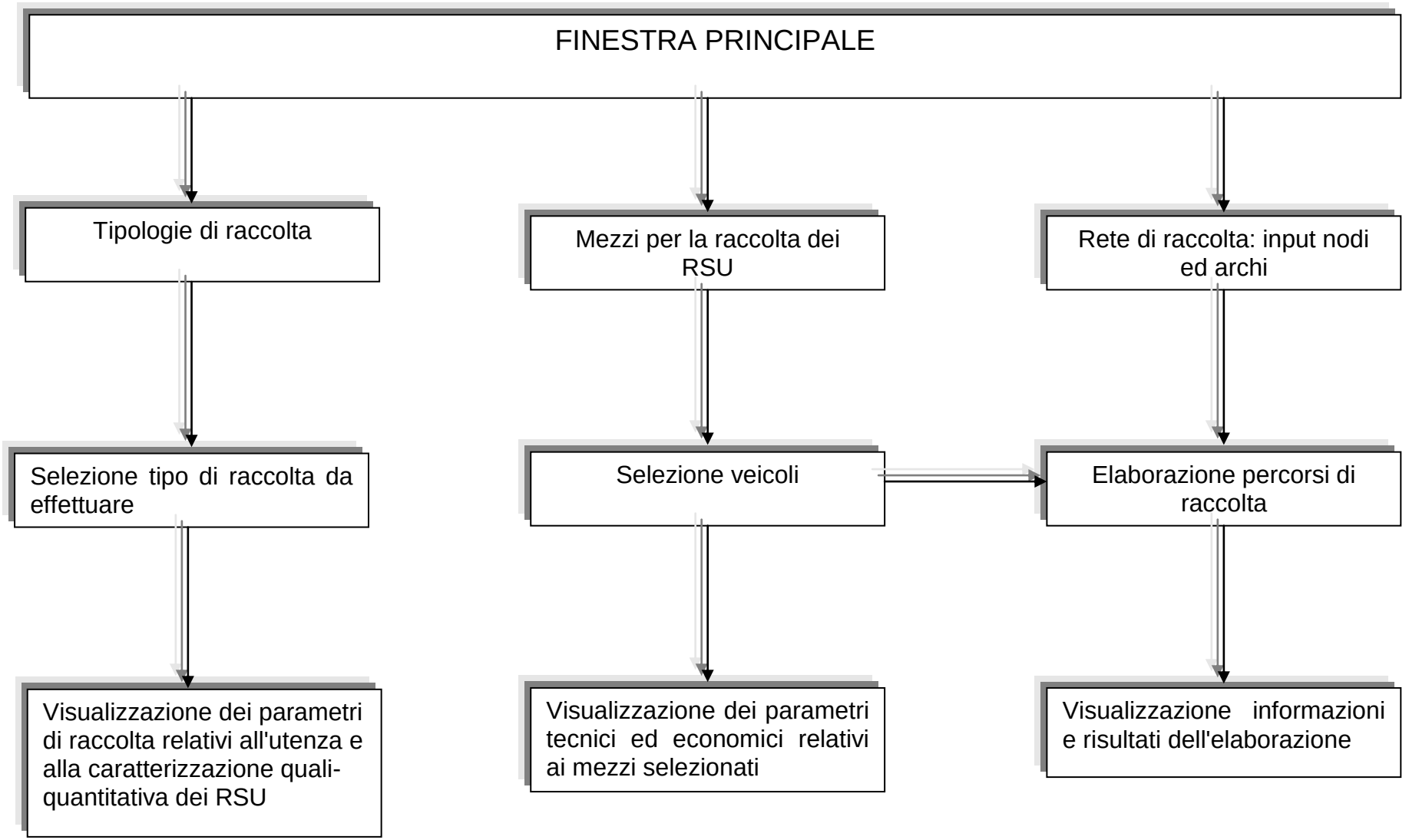
Si è poi deciso di utilizzare come criterio di scelta il *criterio del risparmio* per l'inserimento del nodo nel tragitto parziale.

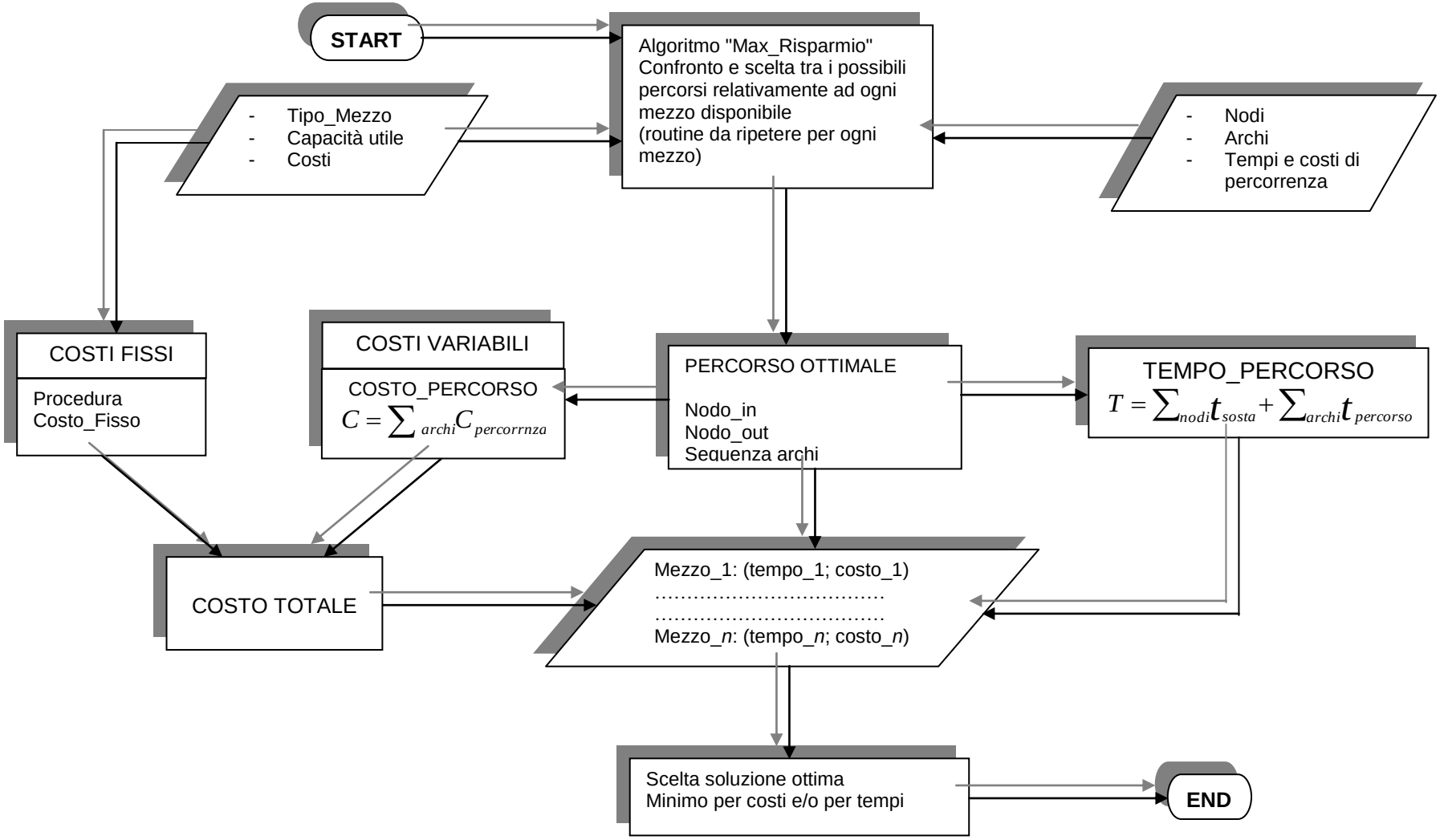
Per i nodi ancora da visitare viene cercata la collocazione all'interno del percorso parziale già definito, inserimento che viene ricercato mediante la *massimizzazione del risparmio*: tra tutti i nodi la cui collocazione è esaminata con la procedura descritta, viene inserito quello che minimizza la funzione obiettivo.
Insieme al nodo viene inclusa nel percorso anche la sequenza di nodi che sono necessari per il collegamento tra il nodo scelto e un estremo del percorso parziale.

La procedura descritta tiene in conto, nelle varie fasi, dei *vincoli dovuti alla viabilità* (archi transitabili solo in determinate fasce orarie, uscite da scuole, uffici, ecc.) e dell'intensità di traffico sull'arco in corrispondenza dei diversi istanti di passaggio del mezzo.

Il diagramma di flusso illustra schematicamente le varie fasi di funzionamento del programma.

SCHEMA GENERALE DI FUNZIONAMENTO DEL PROGRAMMA




START
Algoritmo "Max_Risparmio"
Confronto e scelta tra i possibili percorsi relativamente ad ogni mezzo disponibile
(routine da ripetere per ogni mezzo)
- Tipo_Mezzo
- Capacità utile
- Costi
- Nodi
- Archi
- Tempi e costi di percorrenza
COSTI FISSI
Procedura
Costo_Fisso
COSTI VARIABILI
COSTO_PERCORSO
$C = \sum_{archi} C_{percorrnza}$
PERCORSO OTTIMALE
Nodo_in
Nodo_out
Sequenza archi
TEMPO_PERCORSO
$T = \sum_{nodi} t_{sosta} + \sum_{archi} t_{percorso}$
COSTO TOTALE
Mezzo_1: (tempo_1; costo_1)
……
……
Mezzo_n: (tempo_n; costo_n)
Scelta soluzione ottima
Minimo per costi e/o per tempi
END

Come si è già discusso nei paragrafi precedenti l'approccio fondamentale alla base del prototipo è quello di definire la rete di raccolta con tutte le sue limitazioni e i suoi vincoli.
Tale procedura è favorita dall'interfaccia grafica del programma.
Infatti, è possibile inserire i nodi e gli archi (con le loro proprietà), costituenti la rete attraverso il semplice utilizzo del mouse. Inoltre, l'impostazione dell'*interfaccia grafica* è tale che la visualizzazione dell'oggetto sarà differente a seconda delle sue proprietà.
Prendiamo per esempio l'elemento nodo. Questi avrà *forma e colore differente* a seconda se si tratta di un nodo incrocio, nodo cassonetto per la raccolta indifferenziata, del vetro della carta ecc.
Gli archi saranno tutti orientati quindi un doppio senso sarà rappresentato da due archi paralleli ma con verso opposto. Sarà compito dell'operatore in base ai dati a disposizione dell'azienda preposta alla raccolta definire i tempi di percorrenza, le lunghezze degli archi e il volume di rifiuti associati ad ogni cassonetto.

4. RISULTATI

Gli algoritmi sviluppati hanno come punto di partenza una rete con tutti i vincoli territoriali già immagazzinati nelle proprietà degli elementi che la costituiscono.

La risoluzione della ricerca dei percorsi ottimali è duplice. Come già è stato detto in precedenza nel database i dati possono essere inseriti con due modalità differenti:

- *inserendo i dati relativi ad unico modulo, cioè i dati relativi ad un'unica sottorete la cui capacità complessiva è pari a quella di un solo mezzo;*
- *inserendo i dati relativi ad un'intera zona sulla quale agiscono più camion.*

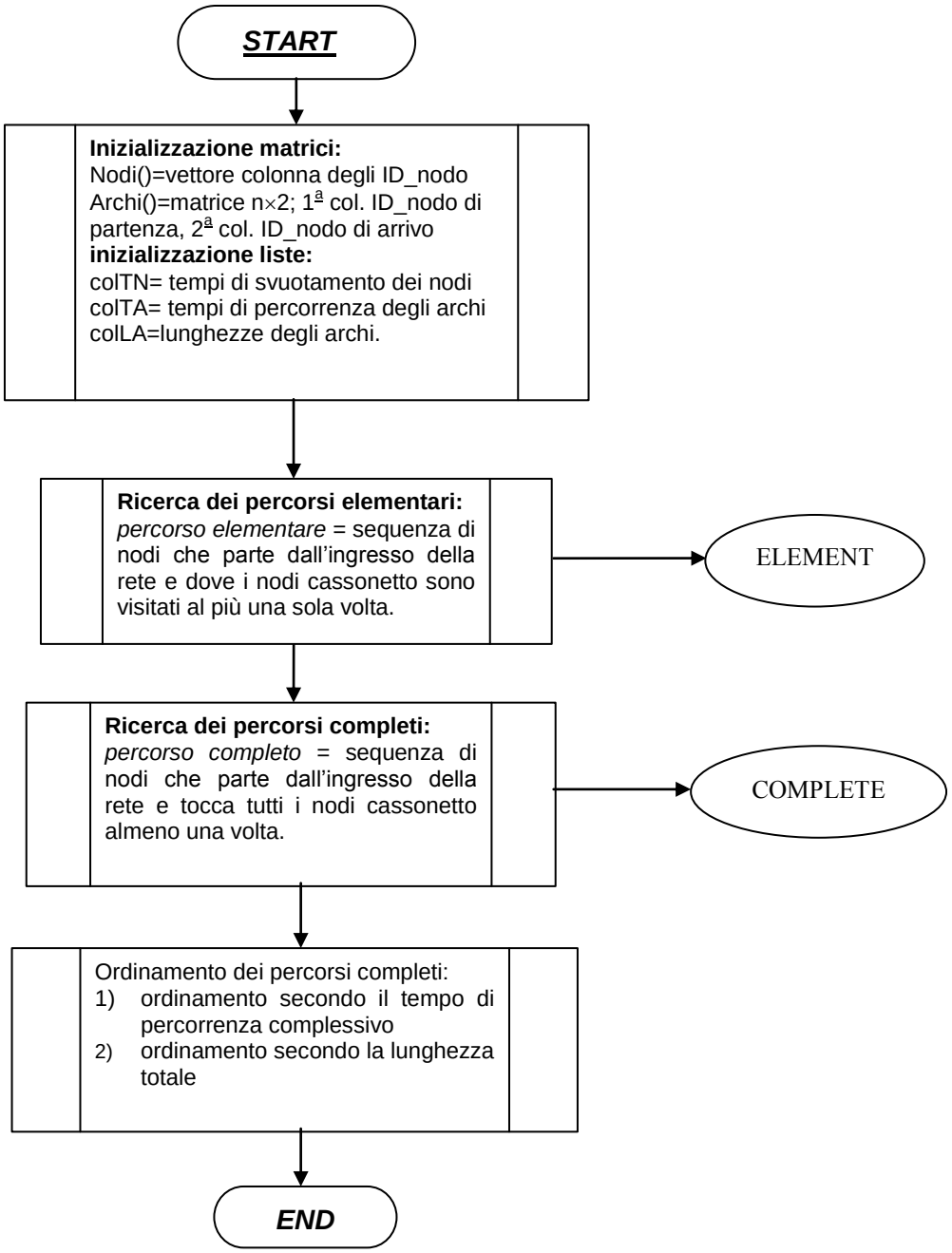
Nel primo caso si utilizzeranno algoritmi di tipo sequenziale che daranno come soluzione una lista di possibili percorsi che toccano tutti i nodi del modulo, ordinati in maniera decrescente secondo la funzione obiettivo.
Nel secondo caso, definito il numero di camion con relativa capacità, attraverso l'algoritmo del *Risparmio,* applicato alla funzione obiettivo, la soluzione sarà costituita da un numero di percorsi pari al numero di camion imposto.
Per ogni percorso viene fornito sia la sequenza dei nodi sui quali il camion transita, sia i nodi che devono essere caricati, sia il tempo di percorrenza impiegato per effettuare l'intero giro.
È possibile che si verifichi la condizione che su un nodo transitino più mezzi a causa della particolare circolazione stradale del tessuto cittadino sotto esame, ma non ci sarà confusione sui nodi che ogni camion deve raccogliere perché l'algoritmo fornisce per ogni mezzo la lista dei nodi ad esso associato. Quindi, nel computo del tempo di percorrenza dell'intero percorso, il tempo di carico e scarico di un nodo sarà associato solo al camion addetto alla sua raccolta.
In tal modo, vengono definiti anche i moduli relativi ad ogni mezzo di raccolta che in questo caso non saranno ben separati tra di loro, ma ci sarà la possibilità che si intersechino lungo la rete stradale.
Passo successivo sarà quello di trovare più percorsi per ogni modulo trovato. Infatti definiti i moduli (cioè i nodi che ogni mezzo deve raccogliere), attraverso l'algoritmo sequenziale sarà possibile trovare una lista di percorsi alternativi a quello trovato con l'algoritmo del *Risparmio* che toccano gli stessi nodi.
L'applicazione delle metodologie euristiche richiede sempre un approccio prudente al sistema in quanto non sempre si è in grado di valutare lo scostamento della soluzione trovata da quella di ottimo e, anche quando si fosse in grado di definire un stima di detto scostamento, non sarebbe possibile avere una stima per ciascuna variabile di scelta utilizzata.
Comunque la rapidità con cui è possibile, una volta definita planimetricamente la zona di operatività del mezzo ed assunte le necessarie informazioni sul traffico e sulla viabilità, ottenere un percorso tendente all'ottimale, consente all'Azienda di disporre uno strumento in grado di effettuare una razionalizzazione del servizio, ed allo stesso tempo un risparmio sui costi e/o un minore impatto del servizio sulla cittadinanza.

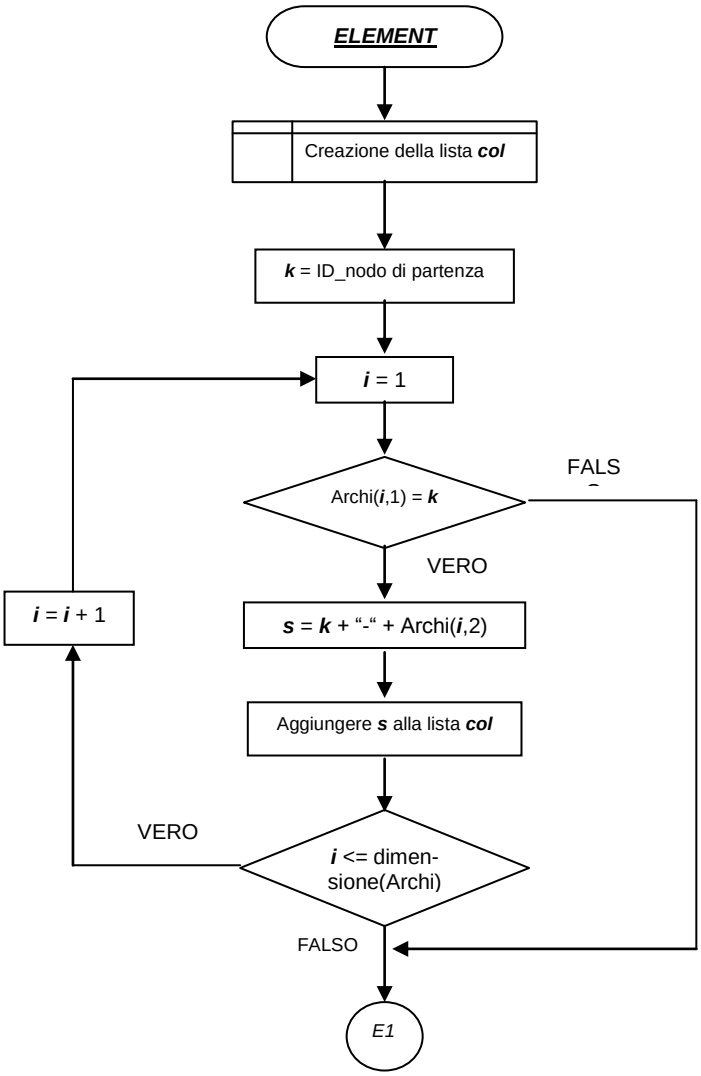
MODELLI DI BASE ADOTTATI NEL PROTOTIPO

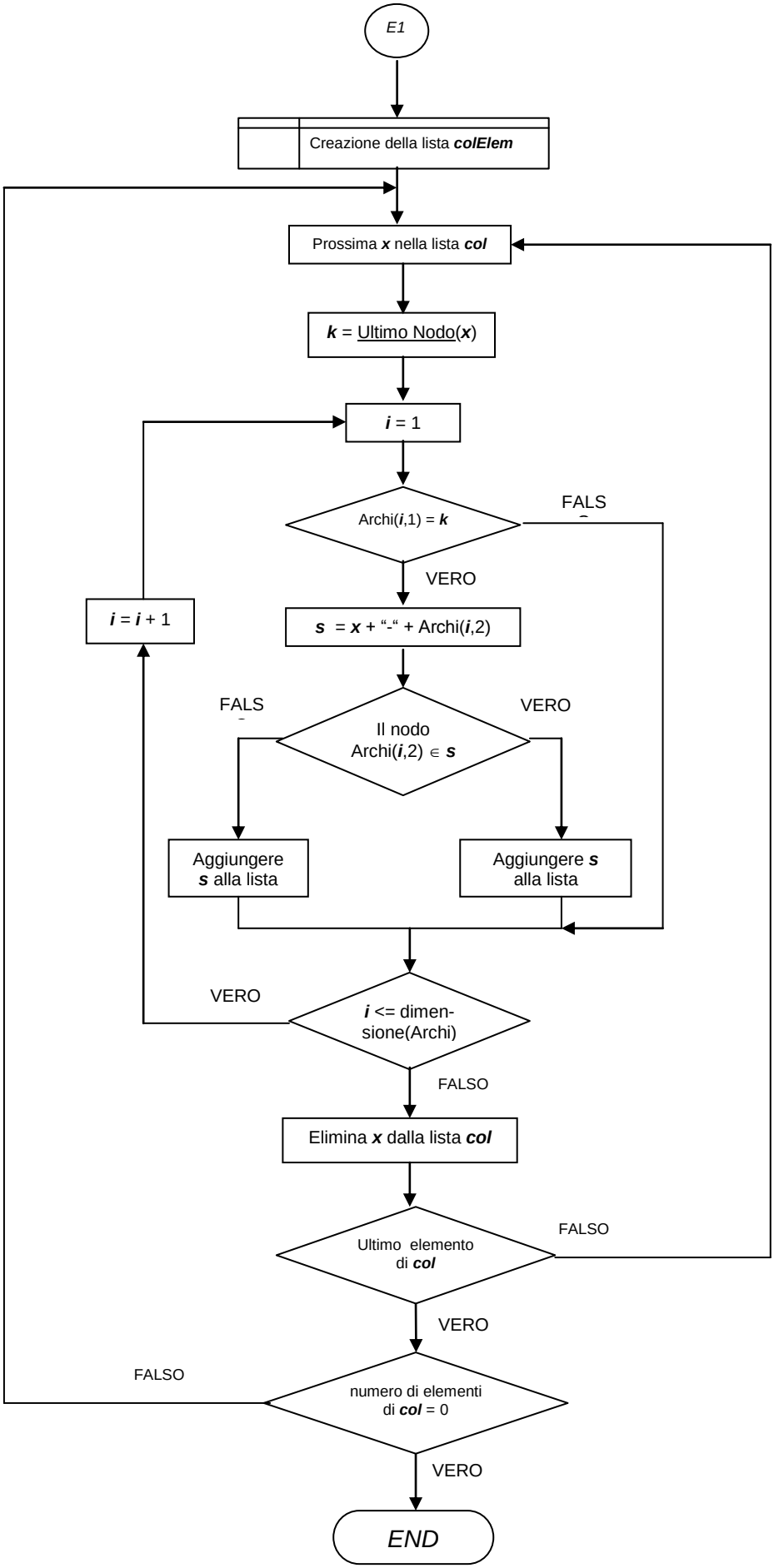
A) MODELLO DEL MINIMO PERCORSO (per modulo/zona - singolo automezzo)

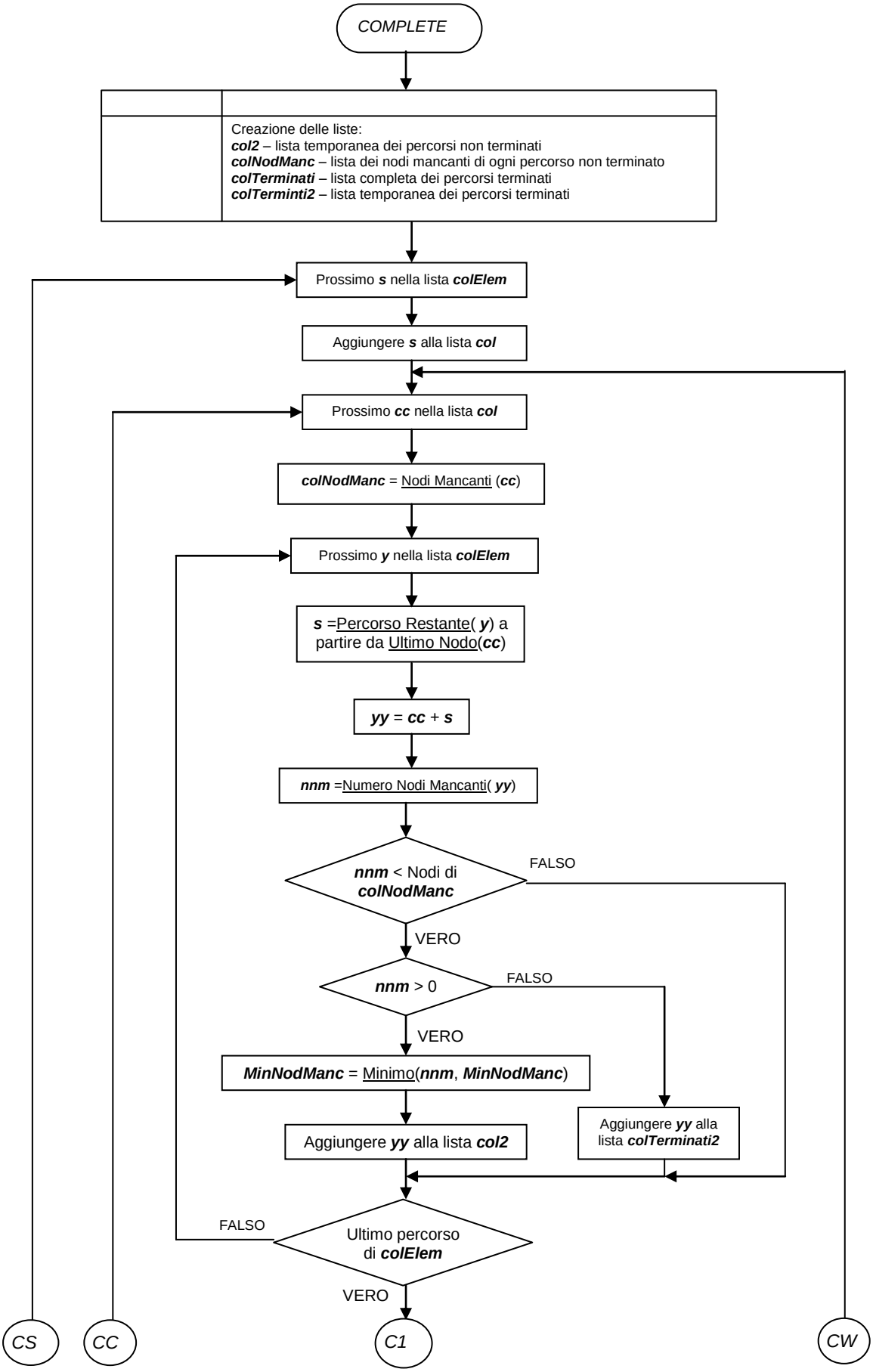
La ricerca del percorso ottimo per tempo o lunghezza, in un "modulo" coincidente con l'area di operatività di un singolo automezzo di differenti capacità, è stata effettuata utilizzando un algoritmo sequenziale. Il problema è: dati *n* cassonetti ed un veicolo raccoglitore, la cui capacità è tale da prelevare tutto il contenuto degli *n* nodi, quest'ultimo deve svuotare tutti i cassonetti fissato il nodo di ingresso e quello di uscita dalla rete, in generale coincidente con il deposito. I passi fondamentali dell'algoritmo sono:

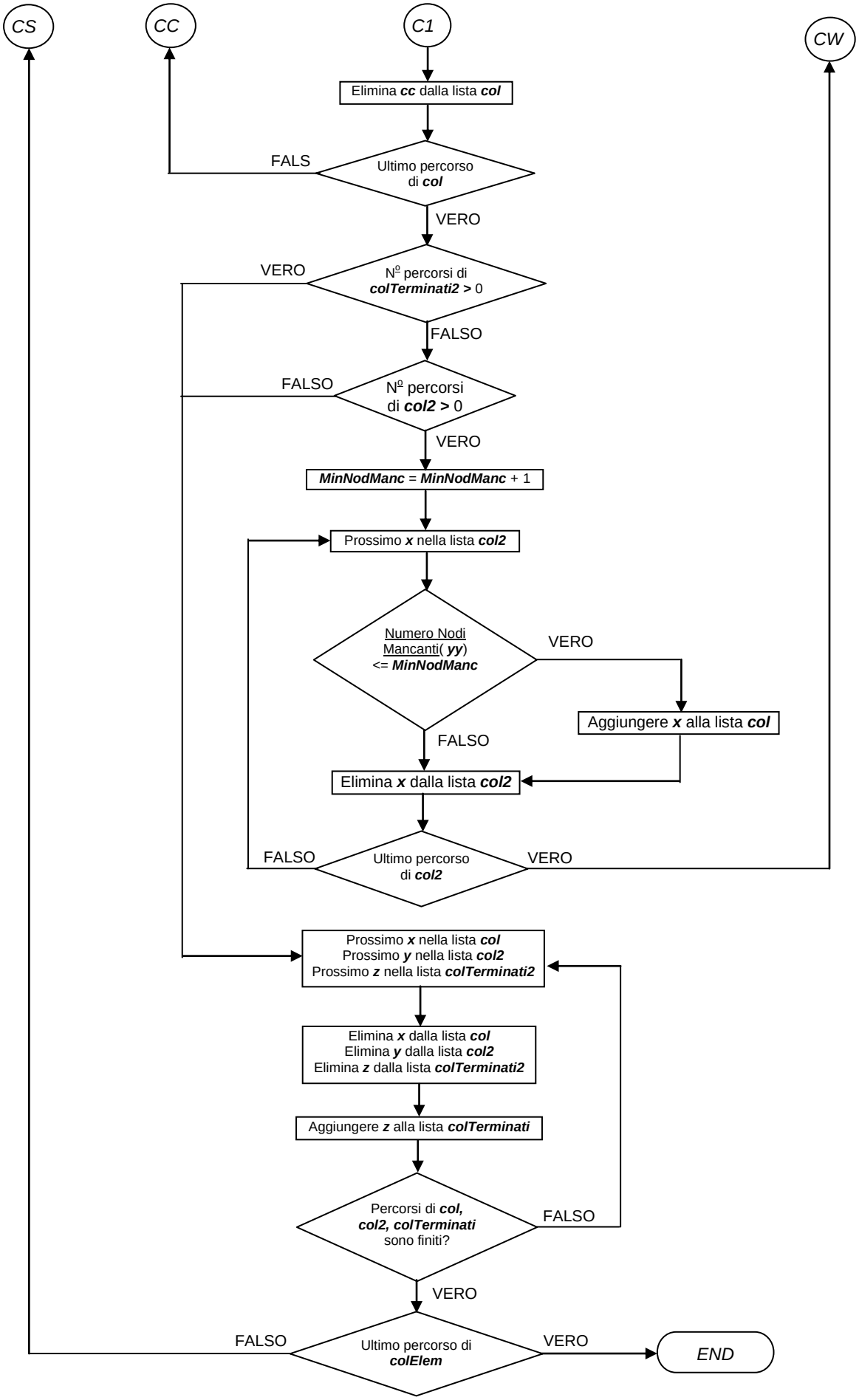
1. *Procedura ELEMENT: definisce tutti i percorsi elementari, rispettando i vincoli della rete viaria;*
2. *Procedura COMPLETE: prende a uno ad uno tutti i percorsi elementari e li compone in sequenza, preferendo quelli che hanno un numero maggiore di nodi mancanti del percorso parziale, e scartando gli altri;*
3. *L'ultima procedura ordina in modo decrescente la lista dei percorsi così ottenuti in base al tempo o alla lunghezza del tragitto.*




ELEMENT
Creazione della lista col
k = ID_nodo di partenza
i = 1
Archi(i,1) = k
FALS
VERO
i = i + 1
s = k + "-" + Archi(i,2)
Aggiungere s alla lista col
VERO
i <= dimen-
sione(Archi)
FALSO
E1


E1
Creazione della lista colElem
Prossima x nella lista col
k = Ultimo Nodo(x)
i = 1
Archi(i,1) = k
FALS
VERO
i = i + 1
s = x + "-" + Archi(i,2)
FALS
Il nodo Archi(i,2) ∈ s
VERO
Aggiungere s alla lista
Aggiungere s alla lista
VERO
i <= dimen-sione(Archi)
FALSO
Elimina x dalla lista col
Ultimo elemento di col
FALSO
VERO
FALSO
numero di elementi di col = 0
VERO
END


COMPLETE
Creazione delle liste:
col2 – lista temporanea dei percorsi non terminati
colNodManc – lista dei nodi mancanti di ogni percorso non terminato
colTerminati – lista completa dei percorsi terminati
colTerminti2 – lista temporanea dei percorsi terminati
Prossimo s nella lista colElem
Aggiungere s alla lista col
Prossimo cc nella lista col
colNodManc = Nodi Mancanti (cc)
Prossimo y nella lista colElem
s =Percorso Restante(y) a partire da Ultimo Nodo(cc)
yy = cc + s
nnm =Numero Nodi Mancanti(yy)
nnm < Nodi di colNodManc
FALSO
VERO
nnm > 0
FALSO
VERO
MinNodManc = Minimo(nnm, MinNodManc)
Aggiungere yy alla lista col2
Aggiungere yy alla lista colTerminati2
Ultimo percorso di colElem
FALSO
VERO
CS
CC
C1
CW


CS
CC
C1
CW
Elimina cc dalla lista col
FALS
Ultimo percorso di col
VERO
VERO
Nº percorsi di colTerminati2 > 0
FALSO
FALSO
Nº percorsi di col2 > 0
VERO
MinNodManc = MinNodManc + 1
Prossimo x nella lista col2
Numero Nodi Mancanti(yy) <= MinNodManc
VERO
Aggiungere x alla lista col
FALSO
Elimina x dalla lista col2
FALSO
Ultimo percorso di col2
VERO
Prossimo x nella lista col
Prossimo y nella lista col2
Prossimo z nella lista colTerminati2
Elimina x dalla lista col
Elimina y dalla lista col2
Elimina z dalla lista colTerminati2
Aggiungere z alla lista colTerminati
Percorsi di col, col2, colTerminati sono finiti?
FALSO
VERO
FALSO
Ultimo percorso di colElem
VERO
END

B) MODELLO EURISTICO DEL MASSIMO RISPARMIO (per zona - più automezzi)

Per la *risoluzione del problema dell'ottimizzazione del percorso con vincoli e con più veicoli* è stato utilizzato un algoritmo che tiene conto dell'effettiva capacità del mezzo preposto alla raccolta.
Si è impiegato il *criterio del risparmio* nella forma da noi modificata per grafi orientati non completamente connessi, per i quali la matrice dei risparmi è in generale non simmetrica. L'algoritmo è caratterizzato da tre procedure principali:

1) Procedura CAMIN: ricerca i percorsi minimi tra due coppie di nodi della rete viaria, implementando l'algoritmo di Floyd-Warshall. Al termine della routine abbiamo inizializzato la matrice m()=[$n \times n$] delle distanze o tempi minimi, e la matrice ms()=[$n \times n$] delle stringhe percorso minimo tra coppie di nodi;

2) Procedura RISP: definisce gli elementi della matrice s()=[$n \times n$] dei risparmi tra coppie di nodi della rete, elementi utilizzati per riempire una lista che viene ordinata in modo decrescente;

3) Procedura MAXRISP: viene scelto il numero e il tipo (volume max) dei mezzi di raccolta; successivamente si ricerca nella lista dei risparmi, a partire dagli archi con risparmio maggiore, i nodi che completano il percorso di ogni singolo automezzo compatibilmente con la sua capacità e ottimizzando rispetto al tempo di percorrenza del tragitto.

L'ultima procedura, che è il nucleo dell'algoritmo di ricerca, merita una descrizione più dettagliata dei passi da compiere:

1. la routine parte scegliendo come arco iniziale quello con il massimo risparmio dalla lista ordinata dalla procedura RISP;

2. si fissa poi la capacità massima dell'automezzo da riempire;

3. dalla lista dei risparmi si estrae, a partire dal massimo risparmio, l'insieme degli archi, tutti con lo stesso risparmio, che possono essere aggiunti all'inizio o alla fine del tragitto parziale;

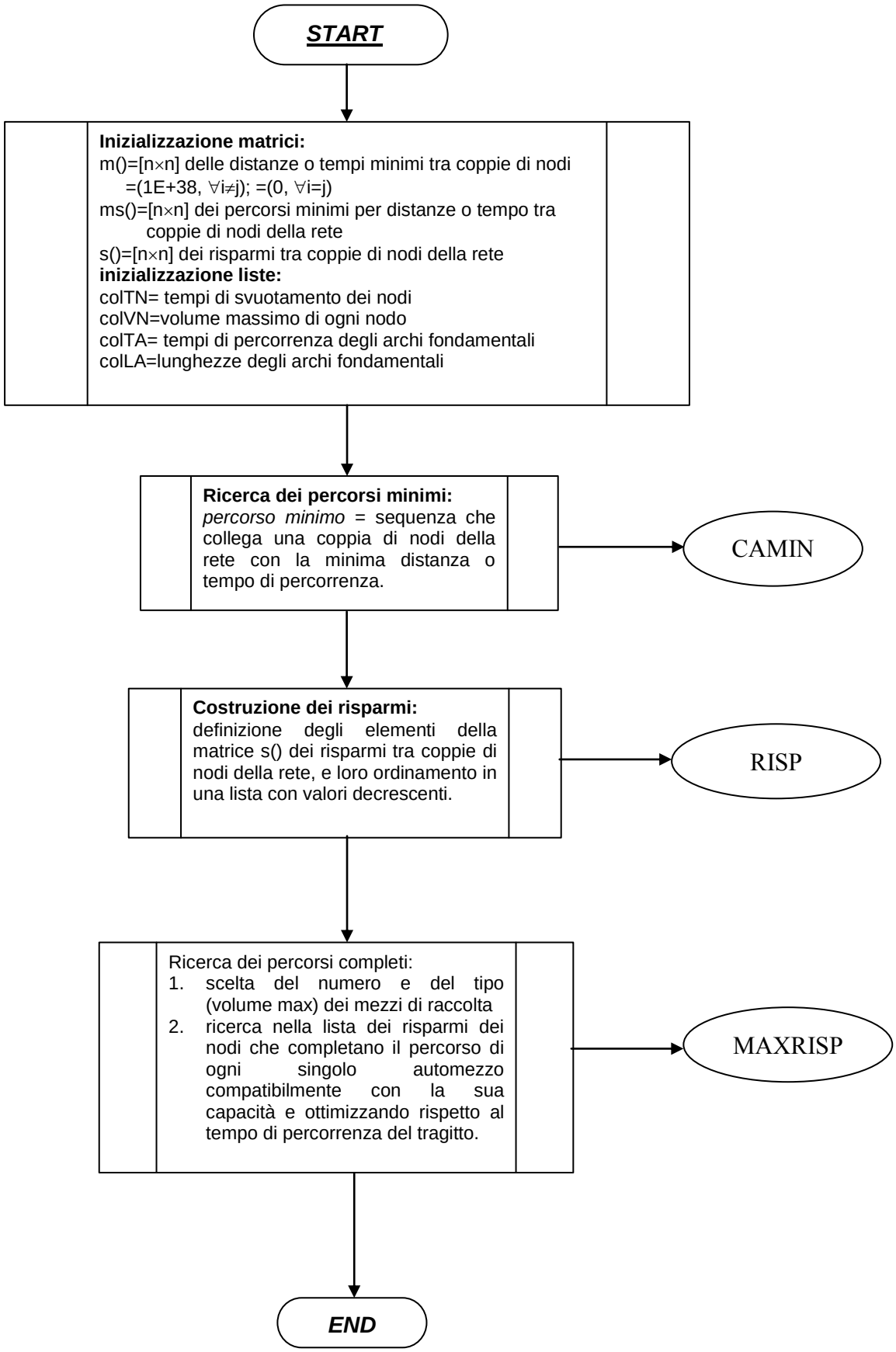
4. dall'insieme degli archi si estrae l'arco che ha il maggior numero di nodi mancanti al percorso parziale, che ha il minor distanza o tempo di percorrenza e che rispetta il vincolo sulla capacità dell'automezzo;

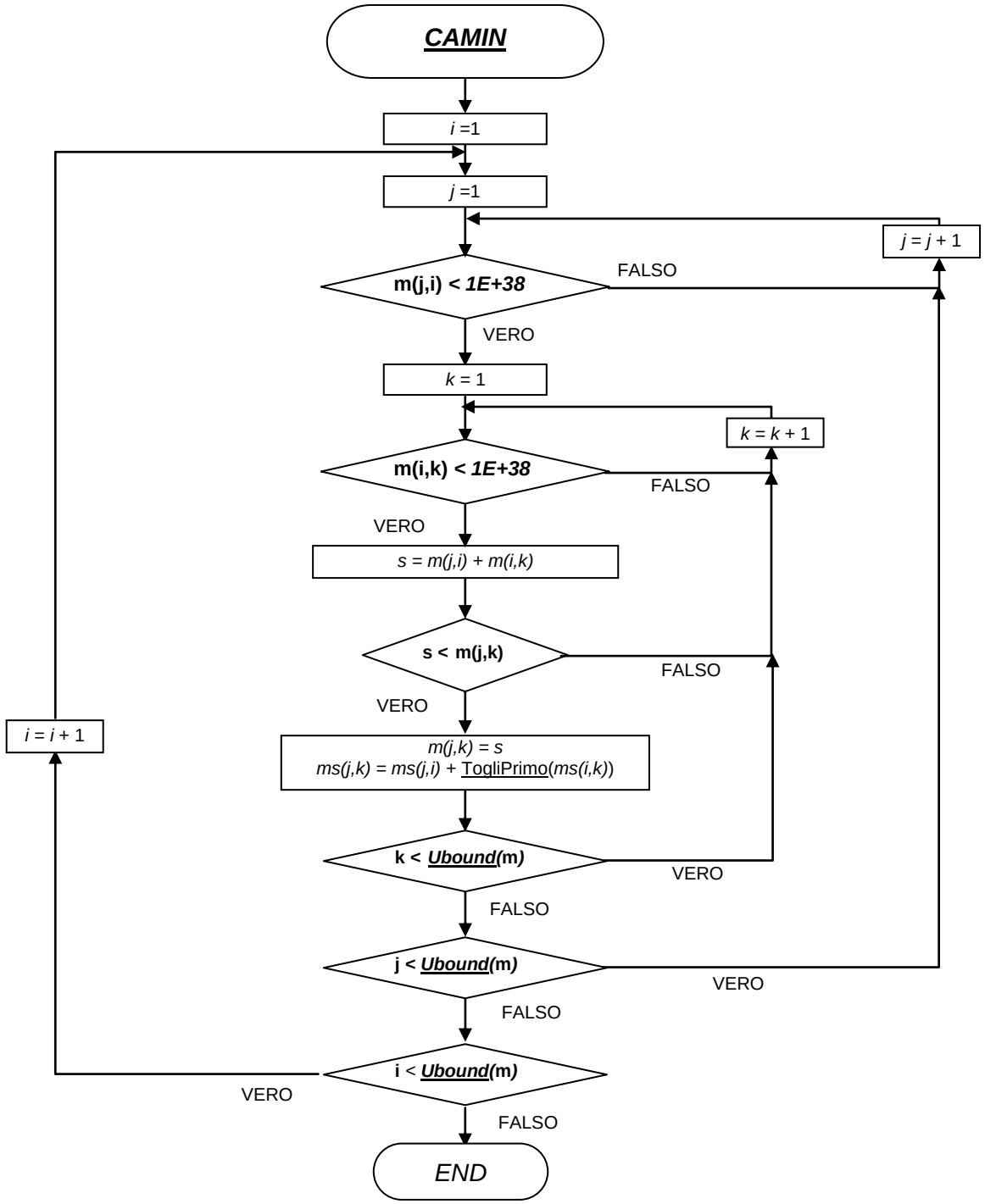
5. l'arco scelto viene aggiunto alla sinistra o alla destra del percorso parziale;

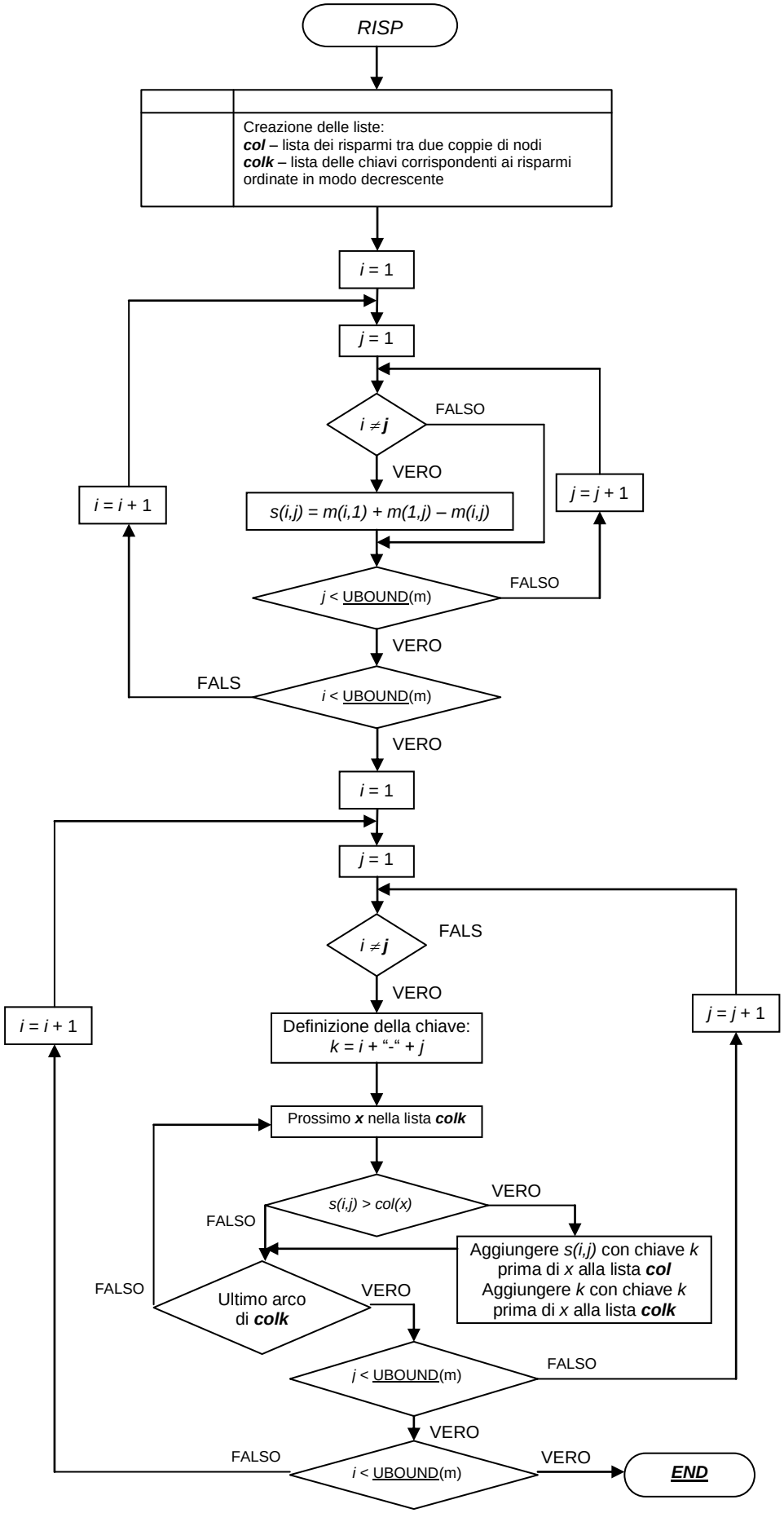
6. se la capacità dell'automezzo è stata raggiunta si esce dal ciclo, altrimenti si ripete dal punto 3;

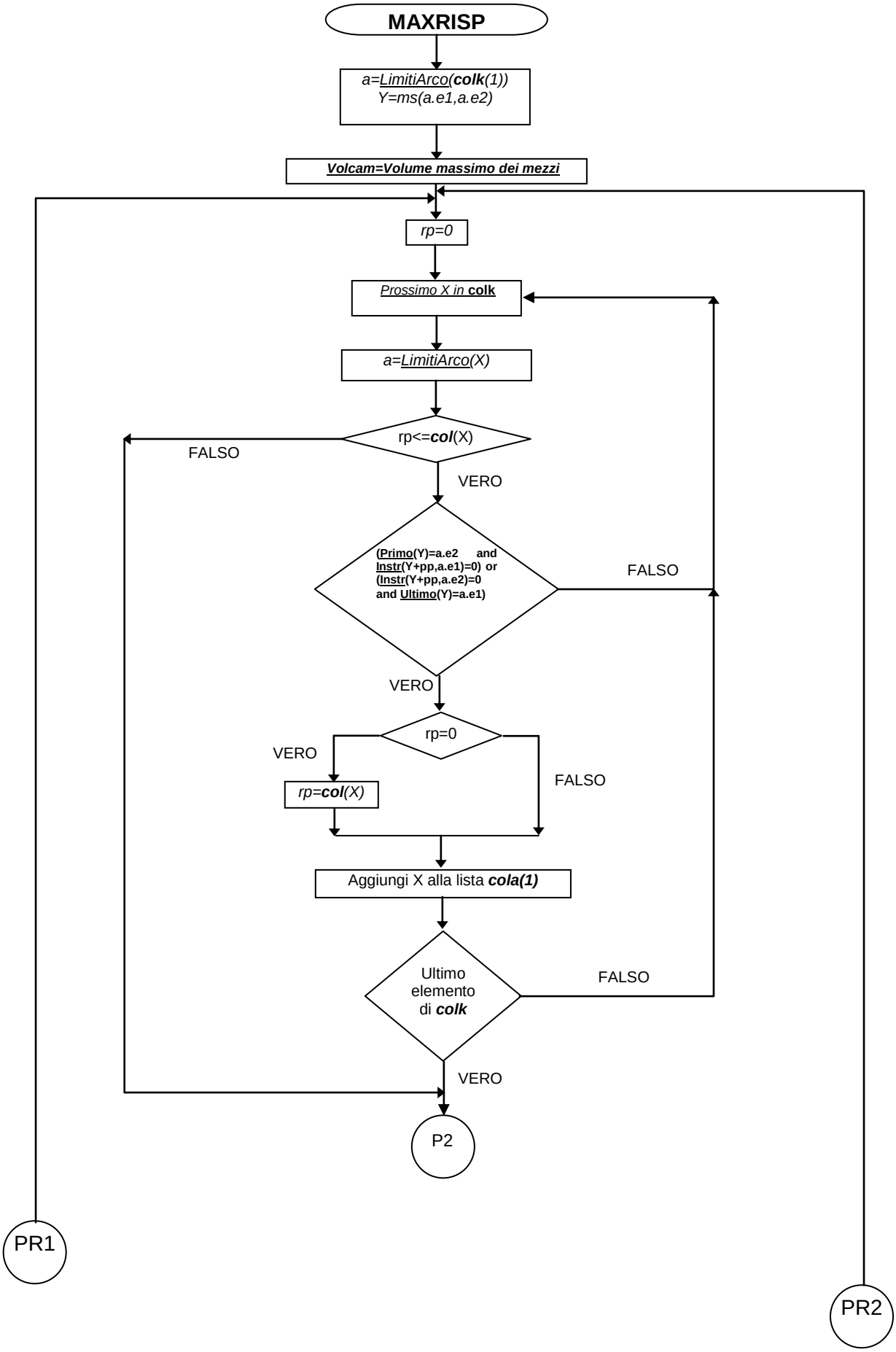
7. viene verificato il numero di nodi visitati dal percorso parziale: se coincide col numero totale dei nodi della rete si esce dall'algoritmo, altrimenti viene creata la lista dei nodi mancanti e viene salvato il primo percorso parziale ottenuto;

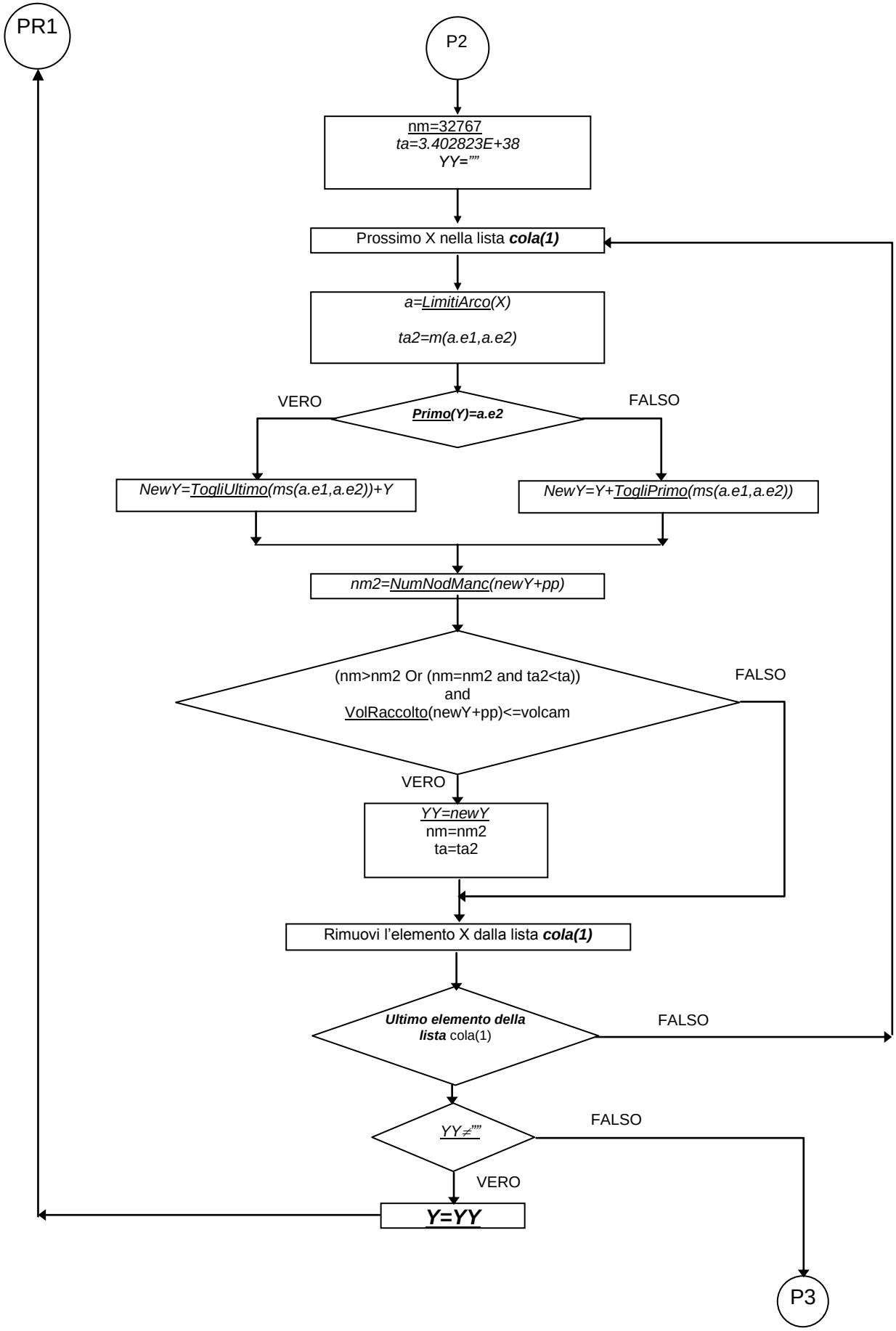
8. si fa ripartire l'algoritmo dal punto 2, scegliendo come arco iniziale il primo arco con il massimo risparmio i cui nodi estremi non sono già contenuti nei percorsi parziali, completi per capacità, finora trovati.

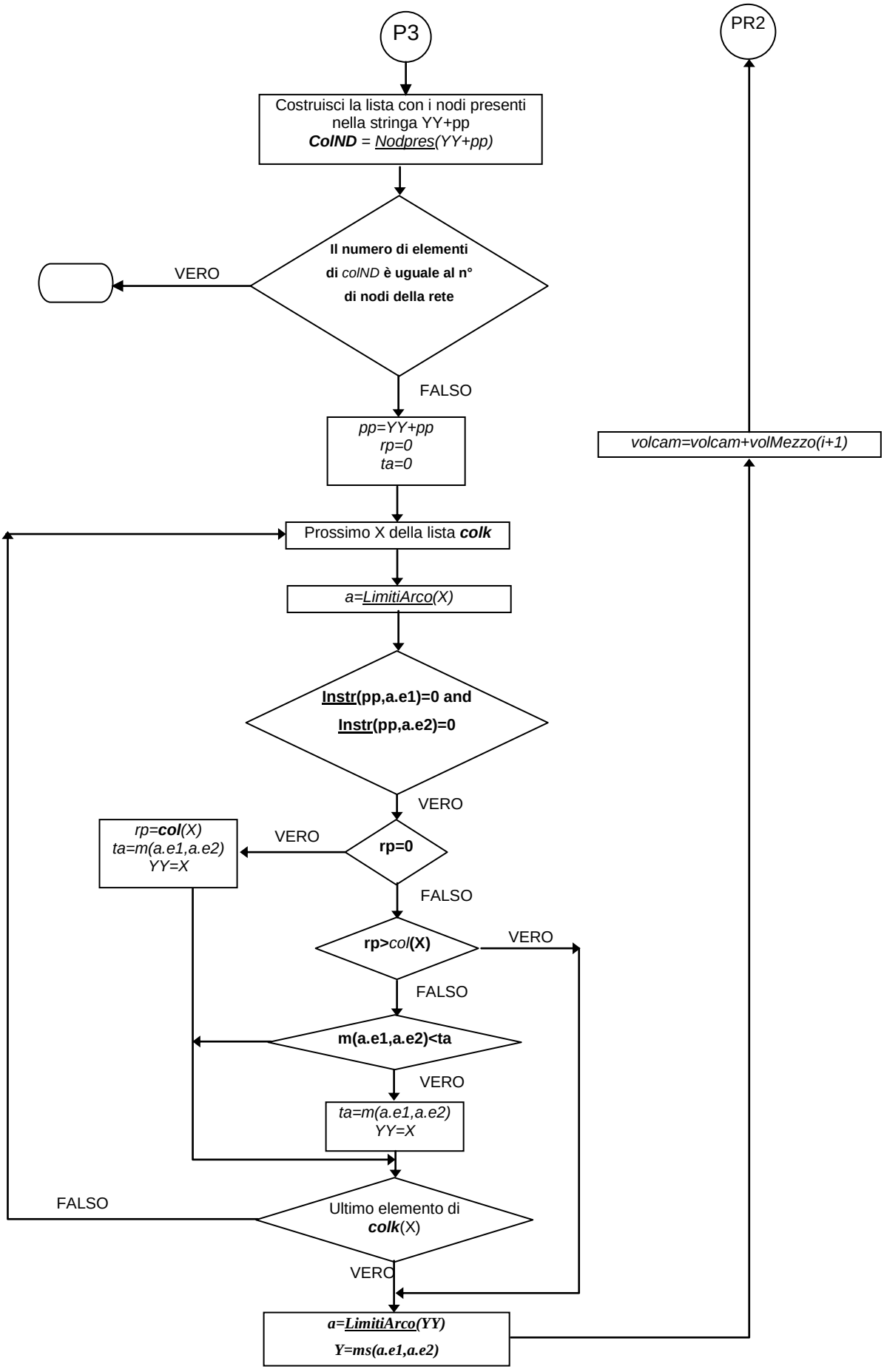

START
Inizializzazione matrici:
m()=[n×n] delle distanze o tempi minimi tra coppie di nodi
=(1E+38, ∀i≠j); =(0, ∀i=j)
ms()=[n×n] dei percorsi minimi per distanze o tempo tra coppie di nodi della rete
s()=[n×n] dei risparmi tra coppie di nodi della rete
inizializzazione liste:
colTN= tempi di svuotamento dei nodi
colVN=volume massimo di ogni nodo
colTA= tempi di percorrenza degli archi fondamentali
colLA=lunghezze degli archi fondamentali
Ricerca dei percorsi minimi:
percorso minimo = sequenza che collega una coppia di nodi della rete con la minima distanza o tempo di percorrenza.
CAMIN
Costruzione dei risparmi:
definizione degli elementi della matrice s() dei risparmi tra coppie di nodi della rete, e loro ordinamento in una lista con valori decrescenti.
RISP
Ricerca dei percorsi completi:
1. scelta del numero e del tipo (volume max) dei mezzi di raccolta
2. ricerca nella lista dei risparmi dei nodi che completano il percorso di ogni singolo automezzo compatibilmente con la sua capacità e ottimizzando rispetto al tempo di percorrenza del tragitto.
MAXRISP
END


CAMIN
i =1
j =1
j = j + 1
m(j,i) < 1E+38
FALSO
VERO
k = 1
k = k + 1
m(i,k) < 1E+38
FALSO
VERO
s = m(j,i) + m(i,k)
s < m(j,k)
FALSO
VERO
i = i + 1
m(j,k) = s
ms(j,k) = ms(j,i) + TogliPrimo(ms(i,k))
k < Ubound(m)
VERO
FALSO
j < Ubound(m)
VERO
FALSO
i < Ubound(m)
VERO
FALSO
END


RISP
Creazione delle liste:
col – lista dei risparmi tra due coppie di nodi
colk – lista delle chiavi corrispondenti ai risparmi ordinate in modo decrescente
i = 1
j = 1
i ≠ j
FALSO
VERO
s(i,j) = m(i,1) + m(1,j) – m(i,j)
j = j + 1
i = i + 1
j < UBOUND(m)
FALSO
VERO
i < UBOUND(m)
FALS
VERO
i = 1
j = 1
i ≠ j
FALS
VERO
Definizione della chiave:
k = i + "-" + j
j = j + 1
i = i + 1
Prossimo x nella lista colk
s(i,j) > col(x)
VERO
FALSO
Aggiungere s(i,j) con chiave k prima di x alla lista col
Aggiungere k con chiave k prima di x alla lista colk
FALSO
Ultimo arco di colk
VERO
j < UBOUND(m)
FALSO
VERO
FALSO
i < UBOUND(m)
VERO
END


MAXRISP
a=LimitiArco(colk(1))
Y=ms(a.e1,a.e2)
Volcam=Volume massimo dei mezzi
rp=0
Prossimo X in colk
a=LimitiArco(X)
rp<=col(X)
FALSO
VERO
(Primo(Y)=a.e2 and Instr(Y+pp,a.e1)=0) or (Instr(Y+pp,a.e2)=0 and Ultimo(Y)=a.e1)
FALSO
VERO
rp=0
VERO
rp=col(X)
FALSO
Aggiungi X alla lista cola(1)
Ultimo elemento di colk
FALSO
VERO
P2
PR1
PR2


PR1
P2
nm=32767
ta=3.402823E+38
YY=""
Prossimo X nella lista cola(1)
a=LimitiArco(X)
ta2=m(a.e1,a.e2)
VERO
Primo(Y)=a.e2
FALSO
NewY=TogliUltimo(ms(a.e1,a.e2))+Y
NewY=Y+TogliPrimo(ms(a.e1,a.e2))
nm2=NumNodManc(newY+pp)
(nm>nm2 Or (nm=nm2 and ta2<ta))
and
VolRaccolto(newY+pp)<=volcam
FALSO
VERO
YY=newY
nm=nm2
ta=ta2
Rimuovi l'elemento X dalla lista cola(1)
Ultimo elemento della lista cola(1)
FALSO
YY≠""
FALSO
VERO
Y=YY
P3


P3
Costruisci la lista con i nodi presenti nella stringa YY+pp
ColND = Nodpres(YY+pp)
Il numero di elementi di colND è uguale al n° di nodi della rete
VERO
FALSO
pp=YY+pp
rp=0
ta=0
Prossimo X della lista colk
a=LimitiArco(X)
Instr(pp,a.e1)=0 and Instr(pp,a.e2)=0
VERO
rp=col(X)
ta=m(a.e1,a.e2)
YY=X
VERO
rp=0
FALSO
rp>col(X)
VERO
FALSO
m(a.e1,a.e2)<ta
VERO
ta=m(a.e1,a.e2)
YY=X
FALSO
Ultimo elemento di colk(X)
VERO
a=LimitiArco(YY)
Y=ms(a.e1,a.e2)
volcam=volcam+volMezzo(i+1)
PR2

C) LISTATI DEI PROGRAMMI DELLE PROCEDURE

```
Dim Nodi() As Long
Dim Archi() As Long
Dim m() As Single
Dim ms() As String
Dim colTN As New Collection 'lista dei tempi di ogni nodo
Dim colVN As New Collection 'lista dei volumi di ogni nodo
Dim colTA As New Collection 'lista dei tempi di ogni arco
Dim colLA As New Collection 'lista delle lunghezze di ogni arco

Private Type arco
   e1 As Long
   e2 As Long
End Type

Private Sub CamminoMinimo() 'algoritmo di Floyd-Warshall
   For i = 1 To UBound(m)
      For j = 1 To UBound(m)
         If m(j, i) < 1E+38 Then
            For k = 1 To UBound(m)
               If m(i, k) < 1E+38 Then
                  s = m(j, i) + m(i, k)
                  If s < m(j, k) Then
                     m(j, k) = s
                     ms(j, k) = ms(j, i) & TogliPrimo(ms(i, k))
                  End If
               End If
            Next k
         End If
      Next j
   Next i
End Sub

Private Function Ultimo(ByVal s As String) As Integer
'Restituisce l'ultimo nodo di una stringa s
   For j = Len(s) To 1 Step -1
      If Mid(s, j, 1) = "-" Then Exit For
   Next j
   Ultimo = Mid(s, j + 1)
End Function

Private Function Primo(ByVal s As String) As Integer
'Restituisce il primo nodo di una stringa s
   For j = 1 To Len(s)
      If Mid(s, j, 1) = "-" Then Exit For
   Next j
   Primo = Mid(s, 1, j - 1)
End Function

Private Function TogliPrimo(ByVal s As String) As String
' Data una stringa s restitusce la parte di stringa successiva al primo nodo
   For j = 1 To Len(s)
      If Mid(s, j, 1) = "-" Then Exit For
   Next j
   TogliPrimo = Mid(s, j)
End Function

Private Function TogliUltimo(ByVal s As String) As String
' Data una stringa s restitusce la parte di stringa precedente l'ultimo nodo
   For j = Len(s) To 1 Step -1
      If Mid(s, j, 1) = "-" Then Exit For
   Next j
   TogliUltimo = Mid(s, 1, j)
End Function

Private Function LimitiArco(ByVal s As String) As arco
```

```
' Data una stringa s restitusce i due nodi dell'arco
   For j = 1 To Len(s)
      If Mid(s, j, 1) = "-" Then Exit For
   Next j
   LimitiArco.e1 = CLng(Mid(s, 1, j - 1))
   LimitiArco.e2 = CLng(Mid(s, j + 1))
End Function

Private Function NumNodManc(ByVal s As String) As Integer
'restitusce il numero di nodi cassonetto mancanti nella stringa s
   NumNodi = UBound(Nodi)
   For i = 1 To NumNodi
      If InStr("-" & s, "-" & Nodi(i) & "-") = 0 Then NumNodManc = NumNodManc + 1
   Next i
End Function

Private Function NodManc(ByVal s As String) As Collection
'restitusce una collection con i nodi cassonetto mancanti ad una stringa
   Dim c As New Collection
   NumNodi = UBound(Nodi)
   For i = 1 To NumNodi
      k = CStr(Nodi(i))
      If InStr("-" & s, "-" & k & "-") = 0 Then
         c.Add k, k
      End If
   Next i
   Set NodManc = c
End Function

Private Function NodPres(ByVal s As String) As Collection
'restitusce una collection con i nodi cassonetto visitati nella stringa s
   Dim c As New Collection
   NumNodi = UBound(Nodi)
   For i = 1 To NumNodi
      k = CStr(Nodi(i))
      If InStr("-" & s & "-", "-" & k & "-") > 0 Then
         c.Add k, k
      End If
   Next i
   Set NodPres = c
End Function

Private Function Cassonetti(ByVal s As String, ByVal p As String) As String
'Data la stringa s del percorso trovato, e la stringa p della somma dei percorsi
'precedenti, restituisce la stringa dei cassonetti da raccogliere separati da virgole
   NumNodi = UBound(Nodi)
   For i = 1 To NumNodi
      k = CStr(Nodi(i))
      If InStr("-" & s & "-", "-" & k & "-") > 0 And InStr("-" & p & "-", "-" & k & "-") = 0 Then
         Cassonetti = Cassonetti & IIf(Cassonetti <> "", ", ", "") & k
      End If
   Next i
End Function

Private Function Rimanente(ByVal s As String, z As String) As String
' Data una stringa s e un nodo z , restitusce la parte di stringa
' successiva al nodo z
   a = InStr("-" & s, "-" & z & "-")
   Rimanente = IIf(a > 0, Right("-" & s, Len(s) - a - Len(z) + 1), "")
End Function

Private Function Tempo(ByVal s As String, Optional cc As Collection) As Single
'Restituisce il tempo impiegato per percorrere una stringa s
   On Error Resume Next
   Dim col As New Collection
   For Each X In cc
      col.Add X, X
   Next X
   For i = 1 To Len(s)
```

```
        If Mid(s, i, 1) = "-" Then
            n1 = IIf(IsEmpty(n2), Mid(s, 1, i - 1), n2)
            For j = i + 1 To Len(s)
                If Mid(s, j, 1) = "-" Then
                    Exit For
                End If
            Next j
            n2 = Mid(s, i + 1, j - (i + 1))
            i = i + Len(n2)

            Call col(n1)
            If Err.Number <> 0 Then
                Err.Clear
                col.Add n1, n1
                Tempo = Tempo + colTN(n1) + colTA(n1 & "-" & n2)
            Else
                Tempo = Tempo + colTA(n1 & "-" & n2)
            End If
        End If
    Next i

    Call col(n2) 'ultimo nodo della stringa
    If Err.Number <> 0 Then Tempo = Tempo + colTN(n2)
End Function

Private Function VolRaccolto(ByVal s As String) As Single
'Restitusce il volume raccolto nel percorso s
    NumNodi = UBound(Nodi)
    For i = 1 To NumNodi
        If InStr("-" & s & "-", "-" & Nodi(i) & "-") > 0 Then _
                    VolRaccolto = VolRaccolto + colVN(CStr(Nodi(i)))
    Next i
End Function

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////  Algoritmo del percorso minimo  //////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

Private Sub cmdApplica_Click()
tmp = Time

'cerchiamo tutti gli archi che partono dal primo nodo Archi(1,1) e li
'aggiungiamo alla collection col; questo ci permette di inizializzare
'la ricerca dei percorsi elementari
    Dim col As New Collection
    k = Archi(1, 1)
    NumArchi = UBound(Archi)
    For i = 1 To NumArchi
        If Archi(i, 1) = k Then
            s = k & "-" & Archi(i, 2)
            col.Add s, s
        Else
            Exit For
        End If
    Next i

'dagli archi di col vengono generati tutti i percorsi elementari sul grafo
'orientato, e sono raccolti nella collection colElem
    Dim colElem As New Collection ' collection dei percorsi elementari
    Do While True
        For Each X In col
            k = Ultimo(X)
            For i = 1 To NumArchi
                If Archi(i, 1) = k Then
                    kbss = X & "-" & Archi(i, 2)
                    If InStr("-" & kbss, "-" & Archi(i, 2) & "-") = 0 Then
                        col.Add kbss, kbss
                    Else
                        colElem.Add kbss, kbss
```

```
                End If
            End If
        Next i
        col.Remove X
    Next X
    If col.Count = 0 Then Exit Do
  Loop
'i = 0
'For Each X In colElem
'    i = i + 1
'    Debug.Print i, X
'Next X

'-------------------------------------------------------------------------------
  On Error Resume Next
  Dim col2 As New Collection
  Dim colNodManc As New Collection
  Dim colTerminati As New Collection
  Dim colTerminati2 As New Collection

ProgressBar1.Max = colElem.Count
ProgressBar1.Min = 1
For k = colElem.Count To 1 Step -1
  s = colElem.Item(k)
  col.Add s, s
  '-------------------------------------------------------------------------
  Do While True
    For Each cc In col
      Set colNodManc = NodManc(cc)
      For Each Y In colElem
        s = Rimanente(Y, Ultimo(cc))
        If s <> "" Then
          YY = cc & s
          nnm = NumNodManc(YY)
          If nnm < colNodManc.Count Then
            If nnm > 0 Then
              MinNodManc = IIf(col2.Count = 0, nnm, Min(nnm, MinNodManc))
              col2.Add YY, YY
            Else
              colTerminati2.Add YY, YY
            End If
          End If
        End If
      Next Y
      col.Remove cc
      If col2.Count > 99 Then Exit For
    Next cc
    If colTerminati2.Count > 0 Then Exit Do

    If col2.Count > 0 Then
      MinNodManc = MinNodManc + 1
      For Each X In col2
        If NumNodManc(X) <= MinNodManc Then col.Add X, X
        col2.Remove X
      Next X
    Else
      Exit Do
    End If

  Loop
  '-------------------------------------------------------------------------
  If col.Count > 0 Then
    For Each X In col
      col.Remove X
    Next X
  End If
  If col2.Count > 0 Then
    For Each X In col2
      col2.Remove X
```

```
      Next X
   End If
   If colTerminati2.Count > 0 Then
      For Each X In colTerminati2
         colTerminati.Add X, X
         colTerminati2.Remove X
      Next X
   End If

'For Each w In colTerminati
'    Debug.Print w
'Next w
'Debug.Print "----------------------------------------"
   ProgressBar1.Value = colElem.Count - k + 1
Next k
'-------------------------------------------------------------------------------

   Open "percorsi.txt" For Output As #1
   Print #1, colTerminati.Count
   For Each z In colTerminati
      Print #1, z
   Next z
   Close #1
   ProgressBar1.Value = 0
   Label1 = Format(Time - tmp, "hh:mm:ss")
   Call RichTextBox1.LoadFile("percorsi.txt")
   RichTextBox1.RightMargin = 1000000000
End Sub

////////////////////////////////////////////////////////  Fine algoritmo del percorso minimo  ////////////////////////////////////////////////////////

Private Sub cmdOrd_Click()
   On Error Resume Next
   Dim colPercOrd As New Collection
   Dim colTempi As New Collection
   Dim colTempi2 As New Collection

   tmp = Time
   Open "percorsi.txt" For Input As #1
   Line Input #1, s
   ProgressBar1.Max = CLng(s)
   ProgressBar1.Min = 1
   While Not EOF(1)
      Line Input #1, s
      t = Tempo(s)
      colTempi.Add t, s
      If colPercOrd.Count = 0 Then
         colTempi2.Add t, CStr(t)
         colPercOrd.Add s, s
      Else
         colTempi2.Add t, CStr(t)
         If Err.Number = 0 Then
            bln = True
            For Each X In colPercOrd
               If t < colTempi(X) Then
                  bln = False
                  colPercOrd.Add s, s, X
                  Exit For
               End If
            Next X
            If bln Then colPercOrd.Add s, s
         Else
            Err.Clear
         End If
      End If
      n = n + 1
      ProgressBar1.Value = n
   Wend
   Close #1
```

```
    Open "percord.txt" For Output As #1
    n = 0
    For Each a In colPercOrd
       n = n + 1
       Print #1, n & ")" & vbTab & "T [min] = " & colTempi(a) & vbTab & a
    Next a
    Close #1

    ProgressBar1.Value = 1
    Label2 = Format(Time - tmp, "hh:mm:ss")
    Call RichTextBox1.LoadFile("percord.txt")
    RichTextBox1.RightMargin = 1000000000
End Sub

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////  Algoritmo del max risparmio  //////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

Private Sub Command1_Click()
    Call CamminoMinimo
    Open "ms.txt" For Output As #1
    For i = 1 To UBound(m)
       For j = 1 To UBound(m)
          Print #1, ms(i, j),
       Next j
       Print #1,
    Next i
    Close #1

    Dim s(23, 23) As Single
    Open "s.txt" For Output As #1
    For i = 1 To 23
       For j = 1 To 23
          If i <> j Then s(i, j) = m(i, 1) + m(1, j) - m(i, j)
          Print #1, s(i, j),
       Next j
       Print #1,
    Next i
    Close #1

    Dim col As New Collection
    Dim colk As New Collection
    For i = 2 To 23
       For j = 1 To 23
          If i <> j Then
             k = i & "-" & j
             If col.Count = 0 Then
                col.Add s(i, j), k
                colk.Add k, k
             Else
                bln = True
                For Each X In colk
                   If s(i, j) > col(X) Then
                      bln = False
                      col.Add s(i, j), k, X
                      colk.Add k, k, X
                      Exit For
                   End If
                Next X
                If bln Then
                   col.Add s(i, j), k
                   colk.Add k, k
                End If
             End If
          End If
       Next j
    Next i
    Open "os.txt" For Output As #1
```

```
        For Each X In colk
            Print #1, X, col(X)
        Next X
    Close #1

    Dim a As arco
    Dim cola() As New Collection
    ReDim cola(1)
    Dim colNd As New Collection
    Dim colTerm As New Collection
    a = LimitiArco(colk(1))
    Y = ms(a.e1, a.e2)
Do While True
    volcam = volcam + 16
    Do While True 'NumNodManc(Y) > 0
        rp = 0 'risparmio
        For Each X In colk
            a = LimitiArco(X)
            If (Primo(Y) = a.e2 And InStr("-" & Y & "-" & pp & "-", "-" & a.e1 & "-") = 0) Or _
                (InStr("-" & Y & "-" & pp & "-", "-" & a.e2 & "-") = 0 And Ultimo(Y) = a.e1) Then
                If rp = 0 Then rp = col(X)
                cola(1).Add X, X
            End If
            If rp > col(X) Then Exit For
        Next X

        nm = 32767 'nodi mancanti -> max integer
        ta = 3.402823E+38 'tempo arco -> max single
        YY = ""
        For Each X In cola(1)
            bln = False
            a = LimitiArco(X)
            newY = IIf(Primo(Y) = a.e2, TogliUltimo(ms(a.e1, a.e2)) & Y, Y & TogliPrimo(ms(a.e1, a.e2)))
            nm2 = NumNodManc(newY & "-" & pp)
            ta2 = m(a.e1, a.e2) 'Tempo(newY, colNd) 'Tempo(ms(a.e1, a.e2))
            If ((nm > nm2) Or (nm = nm2 And ta2 < ta)) And VolRaccolto(newY & "-" & pp) <= volcam Then
                YY = newY
                nm = nm2
                ta = ta2
            End If
            cola(1).Remove X
        Next X
        If YY <> "" Then
            Y = YY
        Else
            Exit Do
        End If
        Debug.Print Y
    Loop

    colTerm.Add Y
    Debug.Print Y, Tempo(Y, colNd), Cassonetti(Y, pp)
    Debug.Print String(25, "/")
    Set colNd = NodPres(Y & "-" & pp)
    If colNd.Count = UBound(Nodi) Then Exit Do
    pp = Y & IIf(Not IsEmpty(pp), "-", "") & pp

    rp = 0
    ta = 0
    For Each X In colk
        a = LimitiArco(X)
        If InStr("-" & pp & "-", "-" & a.e1 & "-") = 0 And InStr("-" & pp & "-", "-" & a.e2 & "-") = 0 Then
            If rp = 0 Then
                rp = col(X)
                ta = m(a.e1, a.e2)
                YY = X
            ElseIf rp > col(X) Then
                Exit For
            Else
```

```
            If m(a.e1, a.e2) < ta Then
               ta = m(a.e1, a.e2)
               YY = X
            End If
         End If
      End If
   Next X
   a = LimitiArco(YY)
   Y = ms(a.e1, a.e2)
Loop
End Sub

////////////////////////////////////////////////////////        Fine algoritmo del max risparmio     ////////////////////////////////////////////////////////

Private Sub Form_Load()
   Dim rsA As New ADODB.Recordset
   Dim rsN As New ADODB.Recordset
   Set rsN = de.rsnodi
   rsN.Open
   'de.cn.Open
   rsA.Open "SELECT ID_nodo1 as N1, ID_nodo2 as N2, tempo as T, lunghezza as L From archi ORDER BY
archi.ID_nodo1, archi.ID_nodo2", de.cn

   ReDim m(rsN.RecordCount, rsN.RecordCount)
   ReDim ms(rsN.RecordCount, rsN.RecordCount)
   For i = 1 To UBound(m)
      For j = 1 To UBound(m)
         m(i, j) = IIf(i = j, 0, 1E+38)
         ms(i, j) = "-"
      Next j
   Next i

   ReDim Nodi(rsN.RecordCount)
   n = 0
   rsN.MoveFirst
   While Not rsN.EOF
      n = n + 1
      Nodi(n) = rsN!ID_nodo
      colTN.Add CSng(rsN!Tempo), CStr(Nodi(n))
      colVN.Add CSng(rsN!volume), CStr(Nodi(n))
      rsN.MoveNext
   Wend
   rsN.Close

   ReDim Archi(rsA.RecordCount, 2)
   n = 0
   rsA.MoveFirst
   While Not rsA.EOF
      n = n + 1
      Archi(n, 1) = rsA!n1
      Archi(n, 2) = rsA!n2
      m(rsA!n1, rsA!n2) = CSng(rsA!t)
      s = CStr(Archi(n, 1)) & "-" & CStr(Archi(n, 2))
      ms(rsA!n1, rsA!n2) = s
      colTA.Add CSng(rsA!t), s
      colLA.Add CSng(rsA!l), s
      rsA.MoveNext
   Wend
   rsA.Close
End Sub

Private Sub RichTextBox1_Click()
   RichTextBox1.RightMargin = 1000000000
End Sub
```