```
function [] = rescue_mission()
        % ====================================================================
        % File name   : rescue_mission.m
        % File Type   : m-file (script file for MatLab)
        % Begin       : 2006-06-23
        % Last Update : 2006-06-24
        % Author      : Nicola Asuni [40552]
        % Description : Tesina di Ottimizzazione
        %               A.A. 2005-2006
        %               Prof. M. Gaviano
        %               UNIVERSITÀ DEGLI STUDI DI CAGLIARI
        %               Facoltà di Scienze Matematiche Fisiche Naturali
        %               Dipartimento di Matematica ed Informatica
        %               Corso di Laurea Specialistica in Tecnologie Informatiche
        % ====================================================================

        % PROGETTO 15 (Missione di soccorso rapido)
        % --------------------------------------------------------------------
        % Il presidente delle Nazioni Unite ha appena ricevuto notizia che in
        % Russia si è verificato un terremoto catastrofico quindi bisogna
        % portare immediato soccorso alla popolazione di tre città; Mosca, San
        % Pietroburgo e Rostov. Gli Stati Uniti hanno messo a disposizione
        % aerei, navi e veicoli per trasportare generi di prima necessità nelle
        % tre città. I mezzi di trasporto sono concentrati a Boston e
        % Jacksonville e hanno le seguenti caratteristiche:
        %
        % +--------------------+------------+----------+----------+
        % | Mezzo di trasporto | nome       | capacità | Velocità |
        % +--------------------+------------+----------+----------+
        % | aereo              | C130       |  150 ton | 400 Km/h |
        % | nave               | nave merci |  240 ton |  35 Km/h |
        % | veicolo            | autocarro  |   16 ton |  60 Km/h |
        % +--------------------+------------+----------+----------+
        %
        % Le navi merci trasporteranno il materiale di soccorso da Boston e
        % Jacksonville ad un porto europeo (Napoli, Amburgo e Rotterdam) e da
        % qui mediante autocarri il materiale è trasportato  nelle tre citta
        % della Russia. Allo stesso modo I C130 devono atterrare in un
        % aeroporto europeo (Londra, Berlino, Istambul) per rifornirsi di
        % carburante prima di raggiungere la Russia. I possibili percorsi sono
        % dati dalla tabella seguente:
        %
        % +--------------+-----------------+----------------+
        % | Da           | A               | Distanza in Km |
        % +--------------+-----------------+----------------+
        % | Boston       | Berlino         |           7250 |
        % | Boston       | Amburgo         |           8250 |
        % | Boston       | Istambul        |           8300 |
        % | Boston       | Londra          |           6200 |
        % | Boston       | Rotterdam       |           6900 |
        % | Boston       | Napoli          |           7950 |
        % | Jacksonville | Berlino         |           9200 |
        % | Jacksonville | Amburgo         |           9800 |
        % | Jacksonville | Istambul        |          10100 |
        % | Jacksonville | Londra          |           7900 |
        % | Jacksonville | Rotterdam       |           8900 |
        % | Jacksonville | Napoli          |           9400 |
        % | Berlino      | San Pietroburgo |           1280 |
        % | Amburgo      | San Pietroburgo |           1880 |
        % | Istambul     | San Pietroburgo |           2040 |
        % | Londra       | San Pietroburgo |           1980 |
        % | Rotterdam    | San Pietroburgo |           2200 |
        % | Napoli       | San Pietroburgo |           2970 |
        % | Berlino      | Mosca           |           1600 |
        % | Amburgo      | Mosca           |           2120 |
        % | Istambul     | Mosca           |           1700 |
        % | Londra       | Mosca           |           2300 |
        % | Rotterdam    | Mosca           |           2450 |
        % | Napoli       | Mosca           |           2890 |
        % | Berlino      | Rostov          |           1730 |
        % | Amburgo      | Rostov          |           2470 |
        % | Istambul     | Rostov          |            990 |
        % | Londra       | Rostov          |           2860 |
        % | Rotterdam    | Rostov          |           2760 |
        % | Napoli       | Rostov          |           2800 |
        % +--------------+-----------------+----------------+
        %
        % Traccia su una mappa la rete dei trasporti associata al problema.
        % Trova i cammini che dovranno seguire i C130, le navi e gli autocarri
        % per giungere a Mosca, San Pietroburgo e Rostov. Nel minor tempo
        % possibile.
        % --------------------------------------------------------------------
```

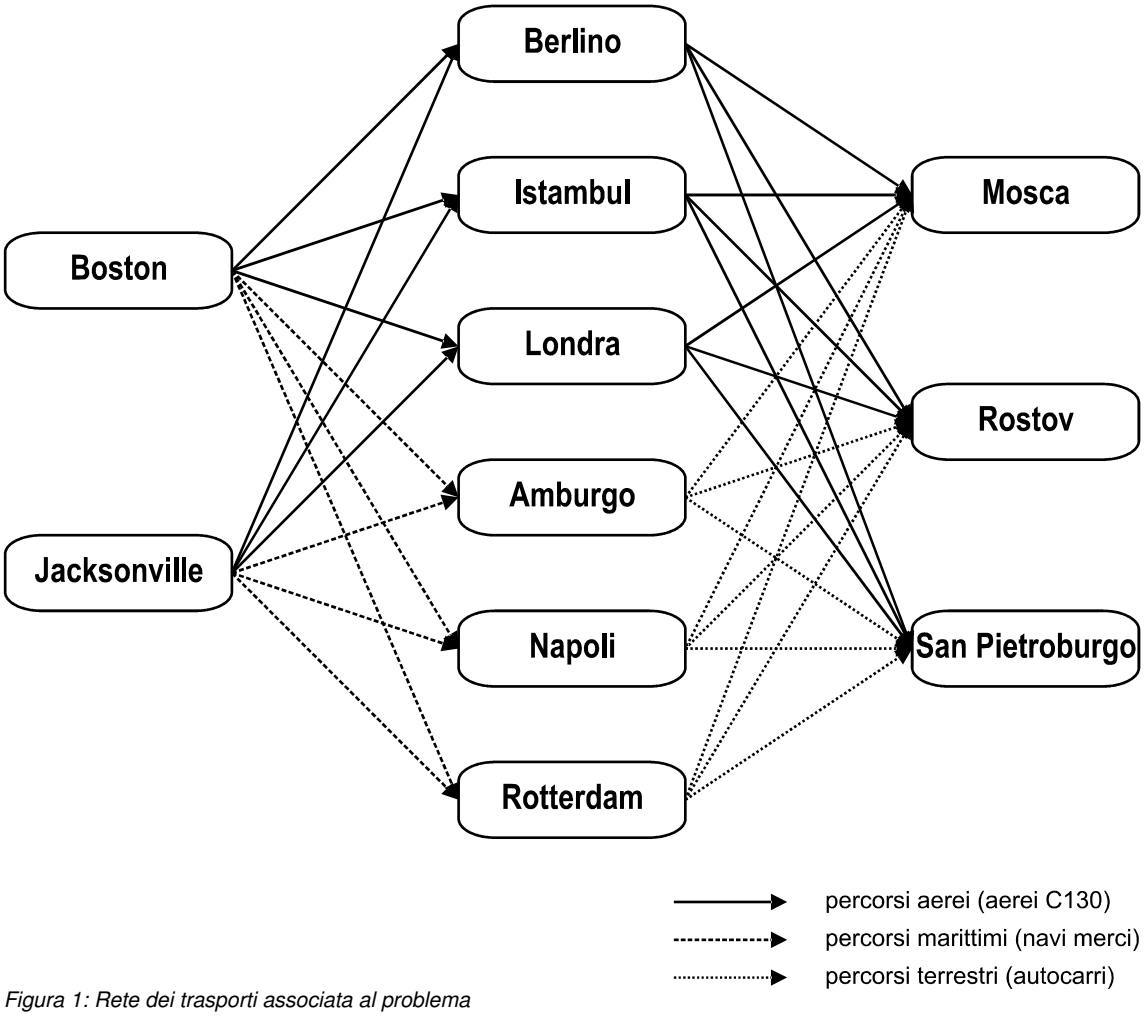

Figura 1: Rete dei trasporti associata al problema

```
% cancello tutte le variabili
clear all;

% nome e tipo dei mezzi di trasporto
mot_names = {
        'aereo', 'C130';
        'nave', 'nave merci';
        'veicolo', 'autocarro';
};

% caratteristiche dei mezzi di trasporto:
% (capacità [tons], velocità [Km/hour])
mot_data = [
        150, 400;
        240, 35;
        16, 60;
];

% nomi delle città
cities = {
        'Amburgo';
        'Berlino';
        'Boston';
        'Istambul';
        'Jacksonville';
        'Londra';
        'Mosca';
        'Napoli';
        'Rostov';
        'Rotterdam';
        'San Pietroburgo';
};

% indici delle città di origine
source_cities = [3;5];

% indici delle città di destinazione
destination_cities = [7;9;11];
```

```
% tabella delle distanze tra città:
% (origine, destinazione, distanza [Km], mezzo di trasporto)
cities_distances = [
        3,  2,  7250, 1;
        3,  1,  8250, 2;
        3,  4,  8300, 1;
        3,  6,  6200, 1;
        3, 10,  6900, 2;
        3,  8,  7950, 2;
        5,  2,  9200, 1;
        5,  1,  9800, 2;
        5,  4, 10100, 1;
        5,  6,  7900, 1;
        5, 10,  8900, 2;
        5,  8,  9400, 2;
        2, 11,  1280, 1;
        1, 11,  1880, 3;
        4, 11,  2040, 1;
        6, 11,  1980, 1;
       10, 11,  2200, 3;
        8, 11,  2970, 3;
        2,  7,  1600, 1;
        1,  7,  2120, 3;
        4,  7,  1700, 1;
        6,  7,  2300, 1;
       10,  7,  2450, 3;
        8,  7,  2890, 3;
        2,  9,  1730, 1;
        1,  9,  2470, 3;
        4,  9,   990, 1;
        6,  9,  2860, 1;
       10,  9,  2760, 3;
        8,  9,  2800, 3
];

% numero delle città (nodi della rete)
[nodes n]= size(cities);

% nodes = 11

% numero delle città di origine (nodi di origine)
[nodes_origin n]= size(source_cities);

% nodes_origin = 2

% numero delle città di destinazione (nodi di destinazione)
[nodes_dest n]= size(destination_cities);

% nodes_dest = 3

% numero di archi (collegamenti diretti tra le città)
[arches n]= size(cities_distances);

% arches = 30

% calcolo i tempi di percorrenza di ogni tratta dividendo la colonna
% delle distanze per la velocità del mezzo che la percorre.
transport_time = (cities_distances(:,3) ./ mot_data(cities_distances(:,4),2));

% transport_time = [18.1250; 235.7143; 20.7500; 15.5000; 197.1429; 227.1429; 23.0000; 280.0000;
  25.2500; 19.7500; 254.2857; 268.5714; 3.2000; 31.3333; 5.1000; 4.9500; 36.6667; 49.5000;
  4.0000; 35.3333; 4.2500; 5.7500; 40.8333; 48.1667; 4.3250; 41.1667; 2.4750; 7.1500; 46.0000;
  46.6667]
```

```
% Ognuna delle 2 città di origine (Boston e Jacksonville) può
% raggiungere ciascuna delle 3 città di destinazione (Mosca, San
% Pietroburgo e Rostov) con 6 possibili cammini, 3 in aereo e 3 in nave
% + autocarro (Figura 2) per un totale di 36 possibili cammini.
% Tra i 3 cammini alternativi tra ogni città di origine ed ogni città
% di destinazione per una data combinazione di mezzi di trasporto,
% dobbiamo scegliere il percorso più breve, quindi in totale cerchiamo
% 12 soluzioni.
```

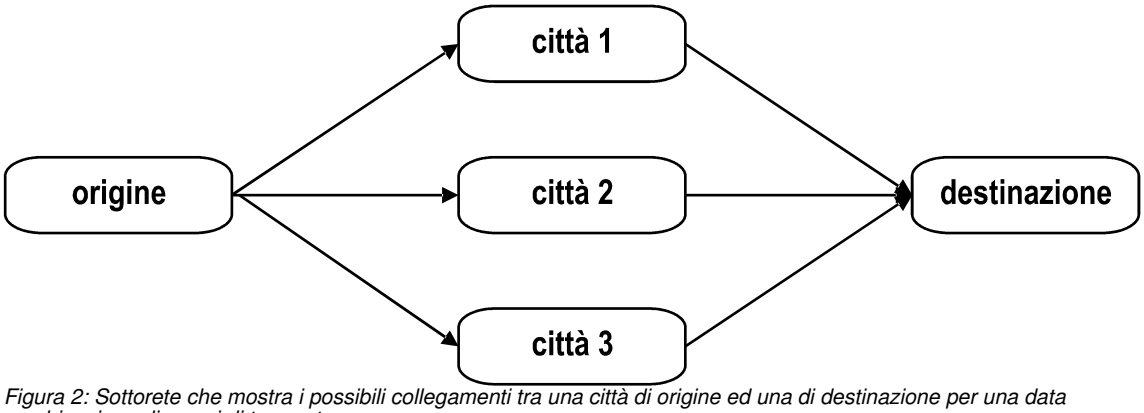

Figura 2: Sottorete che mostra i possibili collegamenti tra una città di origine ed una di destinazione per una data combinazione di mezzi di trasporto

```
% data la particolare geometria della rete utilizzo un algoritmo ad-hoc

% creo una matrice dei tempi di percorrenza con valori infiniti
% indicizzata per (città di origine) x (città di destinazione)
time_matrix = inf.*ones(nodes,nodes);

% inizializzo la matrice dei mezzi di trasporto a zero
% (0=nessun mezzo; 1=aereo; 2=nave; 3=autocarro;)
mot_matrix = zeros(nodes,nodes);

% popolo le matrici
for i=1:arches % scorro la tabella delle distanze
       time_matrix(cities_distances(i,1),cities_distances(i,2)) = transport_time(i,1);
       mot_matrix(cities_distances(i,1),cities_distances(i,2)) = cities_distances(i,4);
end

% time_matrix =
%      Inf      Inf      Inf      Inf      Inf      Inf  35.3333      Inf  41.1667      Inf  31.3333
%      Inf      Inf      Inf      Inf      Inf      Inf   4.0000      Inf   4.3250      Inf   3.2000
% 235.7143  18.1250      Inf  20.7500      Inf  15.5000      Inf 227.1429      Inf 197.1429      Inf
%      Inf      Inf      Inf      Inf      Inf      Inf   4.2500      Inf   2.4750      Inf   5.1000
%  280.0000 23.0000      Inf  25.2500      Inf  19.7500      Inf 268.5714      Inf 254.2857      Inf
%      Inf      Inf      Inf      Inf      Inf      Inf   5.7500      Inf   7.1500      Inf   4.9500
%      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf
%      Inf      Inf      Inf      Inf      Inf      Inf  48.1667      Inf  46.6667      Inf  49.5000
%      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf
%      Inf      Inf      Inf      Inf      Inf      Inf  40.8333      Inf  46.0000      Inf  36.6667
%      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf

% mot_matrix =
%     0     0     0     0     0     0     3     0     3     0     3
%     0     0     0     0     0     0     1     0     1     0     1
%     2     1     0     1     0     1     0     2     0     2     0
%     0     0     0     0     0     0     1     0     1     0     1
%     2     1     0     1     0     1     0     2     0     2     0
%     0     0     0     0     0     0     1     0     1     0     1
%     0     0     0     0     0     0     0     0     0     0     0
%     0     0     0     0     0     0     3     0     3     0     3
%     0     0     0     0     0     0     0     0     0     0     0
%     0     0     0     0     0     0     3     0     3     0     3
%     0     0     0     0     0     0     0     0     0     0     0

% inizializzo le matrici delle soluzioni

% tempi percorsi con aereo
solution_matrix = inf*ones(nodes,nodes);

% tempi percorsi con nave+autocarro
solution_matrix(:,:,2) = inf*ones(nodes,nodes);

% indici dei nodi di transito nei percorsi aerei
solution_matrix(:,:,3) = zeros(nodes,nodes);

% indici dei nodi di transito nei cammini navi+autocarri
solution_matrix(:,:,4) = zeros(nodes,nodes);
```

```
% trovo i percorsi più brevi:

for s=1:nodes_origin % per ogni città di origine
        % ricavo l'indice della città di origine (indice di riga)
        i = source_cities(s);
        for d=1:nodes_dest %per ogni città di destinazione
                % ricavo l'indice della città di destinazione (indice di colonna)
                j = destination_cities(d);
                % (i,j) è la cella di intersezione tra la città di origine e
                % quella di destinazione; calcoliamo quindi il tempo di
                % percorrenza di tutti i possibili cammini e registriamo quello
                % più rapido.
                for k=1:nodes % per ogni possibile nodo di transito (città di transito)
                        % controllo se esiste un cammino dall'origine al nodo di transito
                        if ((k ~= i) && (k ~= j)
                                && isfinite(time_matrix(i,k)) && isfinite(time_matrix(k,j)))
                                % calcolo il tempo totale del cammino tra origine e
                                % destinazione passando per il nodo di transito
                                newtime = time_matrix(i,k) + time_matrix(k,j);
                                % controllo se il tempo è più piccolo di quello
                                % precedentemente trovato
                                if (newtime < solution_matrix(i,j,mot_matrix(i,k)))
                                        % registro il nuovo valore
                                        solution_matrix(i,j,mot_matrix(i,k)) = newtime;
                                        % registro il nodo di transito (città di transito)
                                        solution_matrix(i,j,mot_matrix(i,k)+2) = k;
                                end
                        end
                end
        end
end

% solution_matrix(:,:,1) =
%      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf
%      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf
%      Inf      Inf      Inf      Inf      Inf      Inf  21.2500      Inf  22.4500      Inf  20.4500
%      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf
%      Inf      Inf      Inf      Inf      Inf      Inf  25.5000      Inf  26.9000      Inf  24.7000
%      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf
%      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf
%      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf
%      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf
%      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf

% solution_matrix(:,:,2) =
%      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf
%      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf
%      Inf      Inf      Inf      Inf      Inf      Inf 237.9762      Inf 243.1429      Inf 233.8095
%      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf
%      Inf      Inf      Inf      Inf      Inf      Inf 295.1190      Inf 300.2857      Inf 290.9524
%      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf
%      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf
%      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf
%      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf
%      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf      Inf

% solution_matrix(:,:,3) =
%      0      0      0      0      0      0      0      0      0      0      0
%      0      0      0      0      0      0      0      0      0      0      0
%      0      0      0      0      0      0      6      0      2      0      6
%      0      0      0      0      0      0      0      0      0      0      0
%      0      0      0      0      0      0      6      0      6      0      6
%      0      0      0      0      0      0      0      0      0      0      0
%      0      0      0      0      0      0      0      0      0      0      0
%      0      0      0      0      0      0      0      0      0      0      0
%      0      0      0      0      0      0      0      0      0      0      0
%      0      0      0      0      0      0      0      0      0      0      0

% solution_matrix(:,:,4) =
%      0      0      0      0      0      0      0      0      0      0      0
%      0      0      0      0      0      0      0      0      0      0      0
%      0      0      0      0      0      0     10      0     10      0     10
%      0      0      0      0      0      0      0      0      0      0      0
%      0      0      0      0      0      0     10      0     10      0     10
%      0      0      0      0      0      0      0      0      0      0      0
%      0      0      0      0      0      0      0      0      0      0      0
%      0      0      0      0      0      0      0      0      0      0      0
%      0      0      0      0      0      0      0      0      0      0      0
%      0      0      0      0      0      0      0      0      0      0      0
```

```
% --- STAMPA DEI RISULTATI: ------------------------------------------

fprintf('\nProgetto 15 (missione di soccorso rapido) - Nicola Asuni [40552]\n');
fprintf('\nPERCORSI PIU'' RAPIDI\n\n');

% numera i percorsi trovati
p = 0;

% stampo i risultati
for mot=1:2 % per ogni tipologia di trasporto (1=aereo, 2=nave+autocarro)
        mintime = min(min(solution_matrix(:,:,mot)));
        for s=1:nodes_origin % per ogni città di origine
                % ricavo l'indice della città di origine (indice di riga)
                i = source_cities(s);
                for d=1:nodes_dest %per ogni città di destinazione
                        % ricavo l'indice della città di destinazione (indice di colonna)
                        j = destination_cities(d);
                        if (solution_matrix(i,j,mot+2)>0)
                                % preparo alcune stringhe per la stampa
                                str1 = cities{i};
                                str2 = sprintf('--[%6.2f ore con: %5s]->',
                                        time_matrix(i,solution_matrix(i,j,mot+2)),
                                        mot_names{mot_matrix(i,solution_matrix(i,j,mot+2)),1});
                                str3 = cities{solution_matrix(i,j,mot+2)};
                                str4 = sprintf('--[%6.2f ore con: %7s]->',
                                        time_matrix(solution_matrix(i,j,mot+2),j),
                                        mot_names{mot_matrix(solution_matrix(i,j,mot+2),j),1});
                                str5 = cities{j};
                                str6 = sprintf('%6.2f ore', solution_matrix(i,j,mot));
                                p = p + 1;
                                fprintf('P%02d) %13s %s %9s %s %15s : %s', p, str1, str2, str3,
                                        str4, str5, str6);
                                if (solution_matrix(i,j,mot) == mintime)
                                        % stampo un asterisco per indicare il tempo minimo
                                        % per ogni tipologia di trasporto
                                        fprintf(' *');
                                end
                                fprintf('\n');
                        end
                end
        end
        fprintf('\n');
end
```

```
% Progetto 15 (missione di soccorso rapido) - Nicola Asuni [40552]
%
% PERCORSI PIU' RAPIDI (Figura 3)
%
% P01)       Boston --[ 15.50 ore con: aereo]->    Londra --[ 5.75 ore con:   aereo]->          Mosca :  21.25 ore
% P02)       Boston --[ 18.13 ore con: aereo]->   Berlino --[ 4.33 ore con:   aereo]->         Rostov :  22.45 ore
% P03)       Boston --[ 15.50 ore con: aereo]->    Londra --[ 4.95 ore con:   aereo]-> San Pietroburgo :  20.45 ore *
% P04) Jacksonville --[ 19.75 ore con: aereo]->    Londra --[ 5.75 ore con:   aereo]->          Mosca :  25.50 ore
% P05) Jacksonville --[ 19.75 ore con: aereo]->    Londra --[ 7.15 ore con:   aereo]->         Rostov :  26.90 ore
% P06) Jacksonville --[ 19.75 ore con: aereo]->    Londra --[ 4.95 ore con:   aereo]-> San Pietroburgo :  24.70 ore
%
% P07)       Boston --[197.14 ore con:  nave]-> Rotterdam --[40.83 ore con: veicolo]->          Mosca : 237.98 ore
% P08)       Boston --[197.14 ore con:  nave]-> Rotterdam --[46.00 ore con: veicolo]->         Rostov : 243.14 ore
% P09)       Boston --[197.14 ore con:  nave]-> Rotterdam --[36.67 ore con: veicolo]-> San Pietroburgo : 233.81 ore *
% P10) Jacksonville --[254.29 ore con:  nave]-> Rotterdam --[40.83 ore con: veicolo]->          Mosca : 295.12 ore
% P11) Jacksonville --[254.29 ore con:  nave]-> Rotterdam --[46.00 ore con: veicolo]->         Rostov : 300.29 ore
% P12) Jacksonville --[254.29 ore con:  nave]-> Rotterdam --[36.67 ore con: veicolo]-> San Pietroburgo : 290.95 ore
```

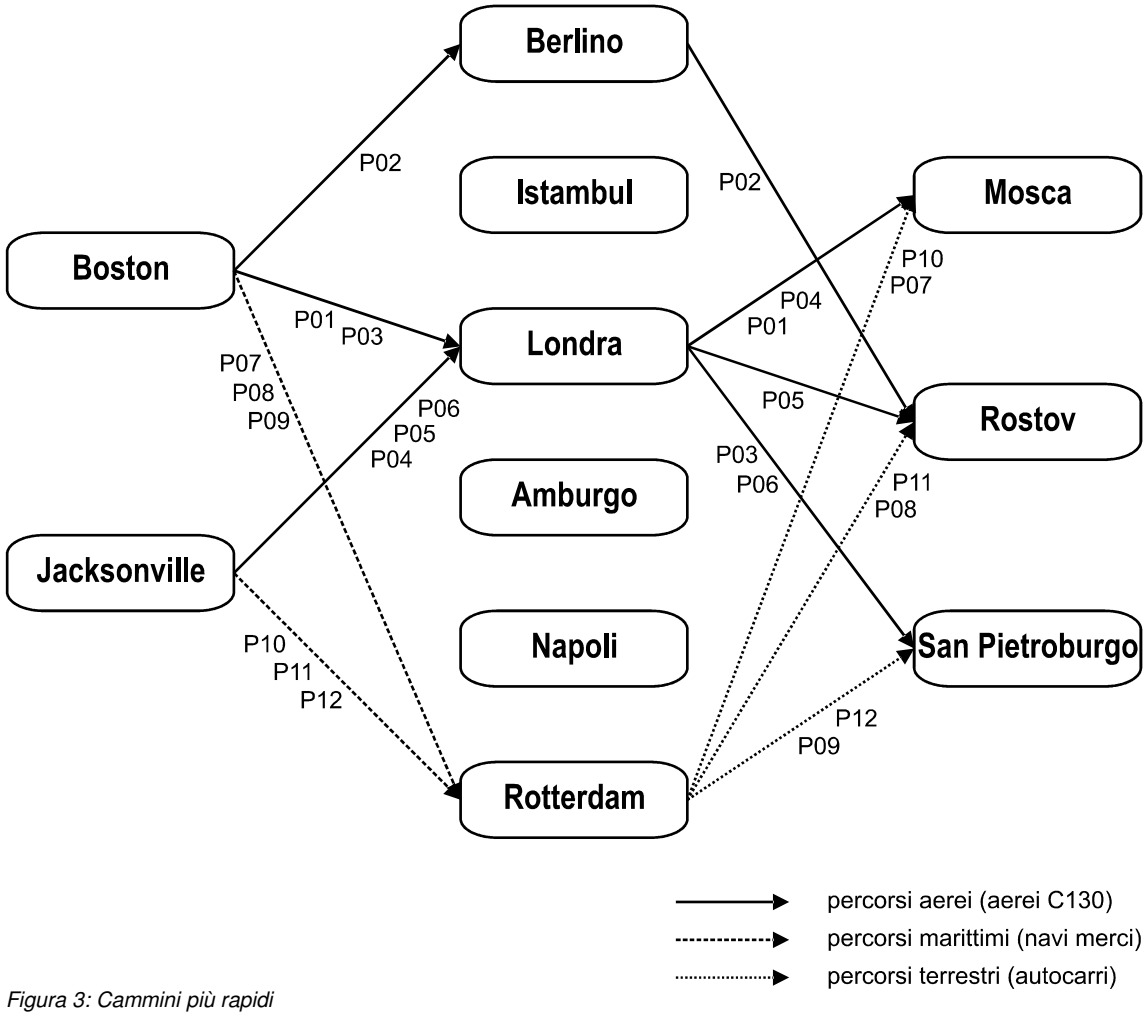

Figura 3: Cammini più rapidi

```
% === EOF ==========================================================
```