Università di Pisa

Dipartimento di Informatica

Technical Report : TR-96-09

Assegnazione dei passeggeri a una rete pubblica interurbana mediante l'enumerazione dei cammini attivi

Pallottino, Stefano; Schettino, Alberta

April 18 1996

ADDR: Corso Italia 40,56125 Pisa,Italy. TEL: +39-50-887111. FAX: +39-50-887226

Assegnazione dei passeggeri a una rete pubblica interurbana

mediante l'enumerazione dei cammini attivi[1]

Stefano Pallottino[2], *Alberta Schettino*[3]

Sommario

Viene descritta l'implementazione di un modello di assegnazione dei passeggeri a una rete di trasporto interurbano con vincoli espliciti di capacità sui veicoli. Inizialmente vengono determinati i cammini attivi rispetto a una soglia ε, detta *minimale*, e successivamente viene assegnata la domanda a detti cammini. Il codice di calcolo descritto determina, oltre alla soglia minimale, anche il flusso di assegnazione.

Introduzione

In questo lavoro viene presentato il codice di calcolo messo a punto per l'implementazione del modello di assegnazione dei passeggeri a una rete di trasporto interurbana proposto in [2, 3].

Nel paragrafo 2, dopo aver brevemente descritto le caratteristiche peculiari della domanda e dell'offerta di trasporto pubblico interurbano, vengono richiamati i concetti essenziali dei modelli di assegnazione dei passeggeri a cammini ε-equivalenti, per le due versioni di costo percepito di cammino descritte in [3], ossia costo generalizzato e disutilità; in particolare viene descritto il modello di assegnazione detto ε-*minimale*. Nel paragrafo 3 vengono proposti tre diversi approcci implementativi per la determinazione del flusso ε-minimale, con relativa descrizione delle procedure utilizzate. Nel paragrafo 4, infine, vengono fornite alcune specifiche tecniche dei codici prodotti e un esempio applicativo.

2. Il trasporto interurbano e l'assegnazione dei passeggeri

2.1 Il servizio di trasporto a orario con capacità

Il servizio è un insieme di linee su gomma o rotaia. Una *linea* è un itinerario che lega tra loro una sequenza di punti geografici, detti *fermate*, in cui i passeggeri possono salire o scendere; a ogni linea è associata almeno una corsa. Una *corsa* è il viaggio di un veicolo lungo la linea, da un capolinea all'altro, secondo una tabella oraria prestabilita che indica per ogni fermata l'orario di arrivo e di partenza. Si assume che gli orari delle corse siano noti agli utenti e che siano effettivamente rispettati. Si conosce inoltre per ogni corsa il numero di posti disponibili sul veicolo, ossia la *capacità* (della corsa o del veicolo). Si assume che tale capacità sia *fissa*, cioè che non sia consentito far salire a bordo più passeggeri del numero di posti disponibili.

2.2 La rete spazio-temporale

Si ipotizza per semplicità che i punti di partenza e di arrivo degli utenti, detti rispettivamente *origine* e *destinazione*, siano raggruppati per zone in punti geografici ideali detti *centroidi*.

---

[1] Lavoro svolto nell'ambito del Progetto Finalizzato Trasporti 2 del C.N.R., progetto no. 93.01669.PF74.

[2] Istituto di Elettronica, Università di Perugia,Via G. Duranti 1/A1,06131 Perugia, e-mail: pallo@istel.ing.unipg.it.

[3] Dipartimento di Informatica, Università di Pisa, Corso Italia 40, 56125 Pisa, e-mail: schettin@di.unipi.it.

Al fine di poter rappresentare le varie corse nella loro evoluzione temporale, il servizio viene descritto mediante una rete *spazio-temporale*, in cui ogni nodo è definito da una coppia (luogo geografico, istante di tempo) e gli archi possono essere di due tipi: quelli spazio-temporali che rappresentano il passaggio da un luogo geografico all'altro nell'intervallo di tempo definito dai nodi estremi dell'arco e quelli solo temporali che collegano lo stesso luogo geografico seppur con tempi diversi. I primi sono costituiti dagli archi *a piedi* che descrivono il movimento tra fermate e centroidi o anche tra fermate geograficamente vicine e dagli archi *a bordo* che descrivono il movimento del veicolo che serve una determinata corsa tra due fermate consecutive secondo i tempi della tabella oraria; i secondi sono gli archi *di salita*, *di discesa*, *di trasbordo* o *di attesa di salita,* necessari a rappresentare i diversi eventi che possono avvenire a una fermata. Per maggiori dettagli si rinvia a [3].

Nel seguito indicheremo con (V,E) la rete spazio-temporale.

2.3 La domanda di trasporto e i "tempi limite"

La *domanda* è definita come numero di utenti che intendono spostarsi col servizio pubblico da un luogo all'altro del territorio nella fascia oraria in esame. A ogni coppia origine/destinazione (o, in breve, o/d) sono associati gli utenti ripartiti in *gruppi*, ciascuno dei quali si distingue dagli altri della medesima coppia o/d per condizioni socio-economiche, motivo dello spostamento e vincoli sui tempi di viaggio. In particolare, a ogni gruppo è associato un *tempo limite* (o *target time*) o all'origine o alla destinazione; nel primo caso consideriamo quegli utenti che possono iniziare il proprio viaggio solo dopo il tempo limite, nel secondo avremo gli utenti che debbono raggiungere la destinazione entro un determinato tempo limite. Modelli di trasporto con scelta dell'orario di partenza oltre che del percorso sono stati studiati in [1, 8, 12].

Oltre ai tempi limite all'origine e alla destinazione, sono previsti anche un *tempo ragionevole di partenza* dall'origine e un *tempo ragionevole di arrivo* alla destinazione: il primo rappresenta, per gli utenti con tempo limite alla destinazione, una soglia di tempo prima della quale essi non sono disposti a partire dall'origine; il secondo rappresenta, per gli utenti con tempo limite all'origine, una soglia di tempo oltre la quale essi non sono disposti a arrivare alla destinazione. Pertanto ciascun utente avrà un *intervallo di tempo* entro il quale effettuare il proprio spostamento, i cui estremi sono i rispettivi tempi limite e ragionevole. Nel seguito indicheremo con *tempo limite* anche il tempo ragionevole quando ciò non comporterà dubbi interpretativi.

Si ipotizza che per ogni gruppo di utenza (coppia o/d con associati tempi limite e ragionevole), che nel seguito indicheremo con k, $k = 1,\ldots, K$, si conosca il numero di persone g_k che intendono viaggiare avvalendosi del servizio pubblico interurbano.

2.4 Il comportamento dei passeggeri

La scelta, per ciascun utente, consiste nel selezionare un *percorso*, cioè una sequenza di spostamenti a piedi e/o a bordo dei veicoli, che inizia all'origine e termina alla destinazione, tra tutti quei percorsi che risultano ammissibili rispetto al proprio intervallo definito dai tempi

limite, cioè quei percorsi che partono dopo il tempo limite all'origine e che arrivano entro il tempo limite a destinazione. Indicheremo con P_k l'insieme di tutti i percorsi ammissibili per il gruppo di utenti k, $k = 1,\ldots, K$.

Una ipotesi di base nei modelli di analisi della mobilità è che gli utenti si comportino in modo razionale, cioè che essi scelgano razionalmente un percorso in funzione del costo o della disutilità associata al percorso stesso. Pertanto si suppone che i passeggeri di uno stesso gruppo *percepiscano* un costo per ciascun percorso per essi disponibile e scelgano il percorso da seguire in base a tale valore *compatibilmente con i posti disponibili sui veicoli*.

Analizziamo due tipi diversi di costo percepito: il primo basato sul concetto di *costo generalizzato* e il secondo sul concetto di *disutilità*. Per maggiori dettagli si rinvia a [3, 11].

2.4.1 Il costo generalizzato

Il concetto di *costo generalizzato* è ampiamente impiegato nella modellistica dei trasporti [6, 7, 9, 10]; nel trasporto interurbano, per ogni percorso p esso può essere definito come la somma ponderata del tempo di viaggio, del tempo globale di attesa e dei ritardi o anticipazioni rispetto ai tempi limite. In questo lavoro definiamo il costo generalizzato di un percorso p come somma ponderata delle seguenti componenti:

- la differenza in_p tra l'orario di partenza del cammino che parte più tardi tra quelli disponibili per il gruppo k e l'orario di partenza del cammino p;
- il tempo globale di attesa w_p nei vari punti di trasbordo del cammino p;
- il numero totale di trasbordi del cammino p, tr_p;
- il tempo f_p impiegato nei tratti a piedi del cammino p;
- il tempo di viaggio a bordo del cammino p, b_p.

Pertanto, dato un percorso p, il suo *costo generalizzato* si ottiene come

$$c(p) = \alpha \cdot in_p + \beta \cdot w_p + \gamma \cdot tr_p + \delta \cdot f_p + \eta \cdot b_p \qquad (2.1)$$

dove α, β, γ, δ e η sono opportuni coefficienti non negativi.

2.4.2 La disutilità

La *disutilità*, così come definita in [3], è una misura della percezione del disagio che l'utente subisce nello scegliere un determinato percorso rispetto alla *scelta ideale*, cioè al percorso che sceglierebbe se tutti i posti disponibili sui veicoli fossero a sua disposizione. Questo disagio non viene percepito dall'utente come una caratteristica "globale" di un dato cammino, ma piuttosto come una sequenza di disagi "locali" associati a ognuno dei momenti in cui egli deve scegliere il percorso da seguire, cioè all'origine, nei nodi di interscambio tra linee e all'arrivo.

La scelta ideale può infatti essere definita per ogni nodo della rete spazio-temporale: se il nodo è l'origine del viaggio avremo un *tempo ideale di partenza*, se è una fermata dove si effettua un trasbordo parleremo di *tempo ideale di prosecuzione del viaggio* (o *di ripartenza*), se il nodo è la destinazione avremo un *tempo ideale di arrivo*. I tempi ideali sono concettualmente diversi a seconda che il tempo limite sia all'origine o alla destinazione.

Vediamo i gruppi di utenti con tempo limite alla destinazione. Per *tempo ideale di partenza* intendiamo il tempo di partenza dall'origine più alto per cui esista un percorso che arrivi a destinazione entro il tempo limite; invece il *tempo ideale di ripartenza* è il tempo di partenza della prima corsa che permetta di proseguire verso la destinazione e di raggiungerla entro il tempo limite. Il *tempo ideale di arrivo* è l'ultimo possibile tempo di arrivo a destinazione non superiore al tempo limite.

Diverso è il caso dei gruppi di utenti con un tempo limite all'origine. Per essi, il *tempo ideale di partenza* è il tempo minimo di partenza all'origine non inferiore al tempo limite e che permette di arrivare a destinazione entro il tempo ragionevole di arrivo; il *tempo ideale di ripartenza* è il tempo di partenza della prima corsa che permetta di proseguire verso la destinazione e di raggiungerla entro il tempo ragionevole. Il *tempo ideale di arrivo* è il tempo minimo possibile di arrivo a destinazione rispettando il tempo limite alla partenza.

Sulla base delle definizioni precedenti, il comportamento degli utenti può essere riassunto nelle regole seguenti:

- *ritardo massimo di partenza* (o *latest departure time*) per i gruppi di passeggeri con tempo limite alla destinazione: nessun utente parte dall'origine prima del necessario per raggiungere la destinazione entro il tempo limite;
- *attesa minima alla partenza* (o *earliest departure time*) per i gruppi di passeggeri con tempo limite all'origine: nessun utente ritarda deliberatamente la partenza dall'origine;
- *attesa minima alle fermate di trasbordo* (o *earliest restarting time*) per tutti i gruppi di utenti: nessun utente ritarda deliberatamente la ripartenza da una fermata di trasbordo.

Possiamo ora definire con maggior precisione la disutilità.

Definizione 2.1
Per ogni gruppo k, $k = 1,\ldots, K$, definiamo *disutilità* (*locale*) all'origine, a ogni punto di trasbordo e alla destinazione rispettivamente il valore assoluto della differenza tra il tempo effettivo di partenza, ripartenza o arrivo e il corrispondente tempo ideale.

Definiamo ora il concetto di disutilità (globale) di un percorso p ammissibile per il gruppo k, che permetta cioè agli utenti del gruppo di spostarsi dall'origine alla destinazione rispettando i vincoli sui tempi limite.

Definizione 2.2
La *disutilità* (*globale*) di un percorso p ammissibile per il gruppo k è il massimo delle disutilità (locali) di p all'origine, a ogni punto di trasbordo e alla destinazione.

2.5 Cammini ε*–equivalenti e flussi* ε*-ammissibili*

L'ipotesi di comportamento razionale alla base dei modelli di analisi della mobilità appare in realtà eccessivamente restrittiva: è infatti regionevole supporre che l'utente, nel confrontare le diverse alternative di percorso, non sia sensibile alla differenza di costo tra il *percorso ideale* e un altro percorso, purché tale differenza non ecceda una determinata soglia di percezione.

Analizziamo più approfonditamente questo aspetto sia per il costo generalizzato che per la disutilità descrivendone le conseguenze in termini di assegnazione dei passeggeri.

2.5.1 ε-equivalenza e costo generalizzato

A ogni gruppo di passeggeri k, $k = 1,\ldots, K$, può essere associato il costo generalizzato del percorso ideale $\gamma_k = \min\{c(p): p \in P_k\}$, cioè il costo del percorso di costo generalizzato minimo. Per ogni percorso ammissibile $p \in P_k$ sia $c_k(p) = c(p)/\gamma_k$ (≥ 1) il *costo relativo* di p.

Sia ε (≥ 1) il valore di *soglia relativa* per cui ogni utente del gruppo k considera equivalente al percorso ideale un qualunque percorso $p \in P_k$ il cui costo relativo $c_k(p)$ sia $\leq \varepsilon$; in tal caso il percorso p è detto ε-*equivalente* al percorso ideale. Indicheremo con $P_k(\varepsilon)$ l'insieme dei cammini ε-*equivalenti* per il gruppo k:

$$P_k(\varepsilon) = \{p \in P_k: c_k(p) \leq \varepsilon\}, \qquad k = 1,\ldots, K.$$

2.5.2 ε-equivalenza e disutilità

In genere sulle disutilità si considera un valore di *soglia assoluta* $\varepsilon \geq 0$: se la disutilità (locale) associata a una determinata scelta alla partenza, in un punto di trasbordo o all'arrivo è non superiore a ε, allora l'utente percepisce tale scelta come indifferente rispetto alla scelta ideale. Un percorso le cui disutilità (locali) siano percepite singolarmente come indifferenti rispetto alle scelte ideali è considerato un percorso ε-*equivalente* al percorso ideale. Anche in questo caso indichiamo con $P_k(\varepsilon)$ l'insieme di tutti i *cammini* ε-*equivalenti* al percorso ideale per il gruppo k, k=1,..., K.

2.5.3 Flussi ε-ammissibili

Sia $G = [g_k]$ la matrice di domanda. Dato un valore di soglia ε, indifferentemente dal modello di comportamento degli utenti scelto (costi generalizzati oppure disutilità), per ogni gruppo k e per ogni cammino $p \in P_k(\varepsilon)$ indicheremo con h_p il *flusso* del cammino p, cioè il numero di passeggeri del gruppo k che scelgono il cammino p. Inoltre ci avvarremo della matrice di incidenza arco-cammino $\Delta=[\delta_{ep}]$, dove $\delta_{ep}=1$ se l'arco e appartiene al cammino p, $\delta_{ep}=0$ altrimenti.

Una assegnazione ε-*ammissibile* è un vettore di flusso $h=[h_p]$, $p \in P_k(\varepsilon)$, $k =1,\ldots, K$, su (V,E) che soddisfa i vincoli della domanda e le capacità sugli archi a bordo; h è detto in tal caso un *flusso* ε-*ammissibile*. L'insieme $F(\varepsilon)$ dei *flussi* ε-*ammissibili* è definito formalmente come segue:

$$\begin{aligned} F(\varepsilon)=\{h:& \\ & \sum_{p \in P_k(\varepsilon)} h_p = g_k, && k=1,\ldots, K, \\ & \sum_{k=1}^{K} \sum_{p \in P_k(\varepsilon)} h_p \delta_{ep} \leq u_e, && \forall\ e \in E_V, \\ & h_p \geq 0, && \forall\ p \in P_k(\varepsilon),\ k=1,\ldots, K\}, \end{aligned} \tag{2.2}$$

dove $E_V \subset E$ è l'insieme degli archi a bordo della rete, e u_e è la capacità dell'arco a bordo $e \in E_V$.

Per maggiori dettagli sulla definizione dei flussi ε-ammissibili si vedano [2,3].

Il problema della determinazione di un flusso ε-ammissibile, conosciuto come *problema di assegnazione*, consiste nell'individuare un flusso $h \in F(\varepsilon)$, se $F(\varepsilon) \neq \emptyset$, oppure segnalare, se $F(\varepsilon)=\emptyset$, che per quel dato valore di soglia ε non esistono flussi ε-ammissibili. La risoluzione di tale problema dipende strettamente dal valore di soglia ε scelto dal momento che l'insieme dei cammini sui quali si possono assegnare le domande dei vari gruppi è proprio l'insieme dei cammini ε-equivalenti.

2.6 Flussi ε-minimali

Un problema più complesso affrontato in [2,3] consiste nel determinare un valore di soglia che sia giustificato dal comportamento degli utenti e per il quale esista un flusso ε-ammissibile. In [3] viene proposto di determinare algoritmicamente un valore di soglia: fornire cioè come soglia, detta *minimale*, il minimo valore di ε per cui esista un flusso ε-ammissibile.

Definizione 2.3

Una soglia $\bar{\varepsilon}$ è detta *minimale* se $F(\bar{\varepsilon}) \neq \emptyset$ e $F(\varepsilon)=\emptyset$, $\forall\ \varepsilon < \bar{\varepsilon}$. Un flusso $h \in F(\bar{\varepsilon})$ è detto un *flusso ε-minimale*.

Sia $[\varepsilon_{min}, \varepsilon_{max}]$ l'intervallo di variazione della soglia ε, dove, per le soglie relative definite mediante il costo generalizzato, ε_{min} (= 1) è il costo relativo del percorso minimo per ciascun gruppo ed ε_{max} (= $\max\{c_k(p)\colon p \in P_k,\ k=1,\ldots,\ K\}$) è il costo relativo massimo dei cammini ammissibili per tutti i gruppi. Nel caso delle disutilità, i cammini ideali per i vari gruppi hanno soglia minima $\varepsilon_{min} = 0$ e ε_{max} è la disutilità massima dei cammini ammissibili per tutti i gruppi.

Senza ledere la generalità supporremo che il problema della determinazione della soglia minimale sia consistente, ossia che $\bar{\varepsilon} \in [\varepsilon_{min}, \varepsilon_{max}]$; inoltre, affinché il problema non sia banalmente risolvibile attraverso l'assegnazione della domanda ai soli cammini ideali, si suppone che $\bar{\varepsilon} > \varepsilon_{min}$.

Sia per i costi generalizzati che per le disutilità il numero di possibili valori di soglia all'interno dell'intervallo $[\varepsilon_{min}, \varepsilon_{max}]$ è un numero finito limitato superiormente dal numero di cammini ammissibili per tutti i gruppi. Indichiamo con $EPS = \{\varepsilon_0, \varepsilon_1,\ldots, \varepsilon_q\}$ *l'insieme dei valori possibili di soglia*, dove $\varepsilon_0 = \varepsilon_{min}$, $\varepsilon_q = \varepsilon_{max}$, e $\varepsilon_i < \varepsilon_{i+1}$, $i = 0, 1,\ldots, q\text{-}1$. Pertanto il problema della determinazione della soglia minimale $\bar{\varepsilon}$ è in realtà un problema di selezione di un elemento, $\bar{\varepsilon}$, da un dato insieme finito, *EPS*.

Nel seguito ci concentreremo essenzialmente sul comportamento degli utenti secondo il costo generalizzato e svilupperemo gli algoritmi per questo caso. I concetti espressi possono essere facilmente adattati al caso delle disutilità e gli algoritmi che ne derivano non differiscono molto da quelli che presenteremo.

3. L'algoritmo di assegnazione ε-minimale

L'individuazione del valore di soglia minimale richiede la risoluzione del corrispondente problema di assegnazione per la quale è necessario disporre di tutti gli insiemi di cammini $\bar{\varepsilon}$-equivalenti, $P_k(\bar{\varepsilon})$, k =1,…, K. Supponiamo di disporre di un valore di soglia, che indicheremo con max_ε, per il quale si è certi che $F(max_\varepsilon)\neq\emptyset$; ovviamente $max_\varepsilon = \varepsilon_l$ per un certo $l \leq q$. Poiché $\bar{\varepsilon}$ potrebbe coincidere con max_ε, l'algoritmo di assegnazione ε-minimale deve essere in grado di enumerare esplicitamente tutti i cammini max_ε-equivalenti per tutti i gruppi di utenti. Il problema della generazione dei cammini max_ε-equivalenti può essere considerato e trattato come un problema a sé stante, in quanto è indipendente dal tipo di approccio algoritmico utilizzato per la determinazione del valore di soglia minimale. In questo paragrafo quindi, analizzeremo innanzitutto tale aspetto, e solo successivamente descriveremo le soluzioni implementative per l'intero problema di assegnazione $\bar{\varepsilon}$-minimale.

3.1 La generazione dei cammini max_ε-equivalenti

Descriviamo ora la generazione dei cammini max_ε-equivalenti per il solo caso dei tempi limite alla destinazione (e tempo ragionevole di partenza arbitrariamente basso); il caso dei tempi limite all'origine richiede solo pochi adattamenti su cui non ci soffermeremo per non appesantire la trattazione algoritmica.

Data la rete spazio-temporale (V, E), con $|V| = n$ e $|E| = m$, indichiamo con N l'*insieme dei nodi fermata* e con C l'*insieme dei nodi centroide*, $N \cup C = V$.

Il generico gruppo di utenti k, k=1,..., K, può essere convenientemente rappresentato tramite la terna $k = (o, d, \tau)$, in cui $d \in C$ è la destinazione del gruppo, $o \in C\backslash\{d\}$ è l'origine, e τ è il tempo limite di arrivo alla destinazione d. Per ogni gruppo k, k = 1,..., K, sia $P_{od\tau} = P_k$ l'insieme dei cammini che partono da o e arrivano a d entro τ.

La fase di generazione dei cammini deve produrre l'insieme $P(max_\varepsilon)$ di tutti i cammini max_ε-equivalenti per tutti i gruppi di passeggeri, definito come:

$$P(max_\varepsilon) = \bigcup_{k=1}^{K} P_k(max_\varepsilon).$$

Nel caso di tempi limite alla destinazione, gli insiemi P_k vengono prodotti a partire dalla destinazione d seguendo l'ordine spazio-temporale inverso a quello naturale, in modo tale da evitare la generazione anche di quei cammini che arrivano alla destinazione oltre il tempo limite.

Inoltre, i cammini dei vari gruppi aventi destinazione comune vengono prodotti congiuntamente. Indicando con θ il massimo valore di tempo limite alla destinazione, per ogni destinazione $d \in C$ viene costruito l'insieme $P_{d\theta}$ di tutti i cammini che partono da un qualsiasi nodo della rete e arrivano alla destinazione d entro θ. Poiché

$$P_{od\tau} \subseteq P_{d\theta} \qquad \forall\, o \in C\backslash\{d\}, \forall\, \tau \leq \theta,$$

una volta costruito $P_{d\theta}$, la determinazione dei singoli insiemi $P_{od\tau}$ consiste semplicemente nella selezione da $P_{d\theta}$ del sottoinsieme di cammini che partono da o ed arrivano a d entro τ.

Nel seguito descriveremo la fase di generazione degli insiemi $P_{d\theta}$ e successivamente vedremo la generazione dell'insieme $P(max_\varepsilon)$ nel suo complesso.

3.1.1 La generazione di $P_{d\theta}$

Nel paragrafo 2.4 abbiamo indicato con P_k l' insieme di tutti i percorsi ammissibili per il gruppo di utenti k, $k = 1,\ldots, K$, che corrisponde all'insieme delle alternative di scelta per il gruppo k. Il concetto di *ammissibilità* di un cammino merita in questa sede di essere maggiormente approfondito. In [4, 5] è stata posta in rilievo l'importanza della corretta definizione, nei modelli di scelta del percorso, dei percorsi ammissibili. Nel modello che presentiamo, in cui si enumerano esplicitamente i percorsi, è importante applicare criteri selettivi dei percorsi *ammissibili* dall'insieme dei percorsi *elementari*, criteri che da una parte siano giustificati dal comportamento degli utenti, e che dall'altra permettano di limitare sensibilmente il numero di percorsi generati.

Esemplificando, consideriamo un cammino *p elementare* per il gruppo $k = (o, d, \tau)$, ossia una sequenza di archi a piedi e a bordo su (V, E) che parta da o e arrivi in d entro τ; un cammino così definito può senz'altro contenere due archi consecutivi appartenenti a corse diverse della stessa linea, ma nessun utente razionale scenderebbe da un autobus a una fermata per aspettare l'autobus successivo della stessa linea.

Prendendo spunto da considerazioni di questo tipo, perveniamo alla seguente definizione di cammino ammissibile:

Definizione 3.1

Un cammino elementare p per il gruppo k, k=1,…, K, è *ammissibile* se:

- non contiene cicli nel grafo spaziale, ossia se la sequenza di fermate attraversate non contiene più di una volta lo stesso nodo;
- utilizza coerentemente le linee e le corse, cioè:
    - non contiene archi a bordo consecutivi della stessa linea ma di corse diverse;
    - non contiene archi non consecutivi della stessa linea.

La fase di costruzione dell'insieme $P_{d\theta}$, effettuata dalla procedura **Build**, produce l'insieme di tutti i cammini che partono da un qualsiasi nodo e arrivano alla destinazione d entro θ, e che siano ammissibili in base alla definizione 3.1. Per ragioni di semplicità espositiva, nella seguente descrizione di questa fase utilizzeremo il termine *cammino* sottointendendo che esso sia ammissibile.

Indichiamo con $FS(i)$ e con $BS(i)$ rispettivamente la *stella uscente*, o *forward star*, e la *stella entrante*, o *backward star*, del nodo $i \in V$:

$$FS(i) = \{(u, v) \in E : u = i\}; \quad BS(i) = \{(u, v) \in E : v = i\}.$$

Dati la destinazione $d \in C$ e un nodo $i \in V \setminus \{d\}$, sia $e = (i, t_i, j, t_j) \in FS(i)$ un arco della stella uscente di i; e è detto *arco utile per i verso d* se esiste almeno un cammino p che parta da i e arrivi a d entro θ, il cui primo arco sia e. Sia $U_d(i) \subseteq FS(i)$ *l'insieme degli archi utili per i verso d*:

$U_d(i) = \{e \in FS(i)$: e è utile per i verso $d\}$.

Per ogni $e = (i, t_i, j, t_j) \in U_d(i)$ definiamo *insieme attivo dell'arco e verso d*, e lo indicheremo con $ACT_d(e)$, l'insieme di tutti i cammini $\{e\} \cup q$, ottenibili per *concatenazione* di e con cammini q che iniziano a j e terminano a d, ammissibili per θ:

$ACT_d(e) = \{p: p = \{e\} \cup q$, q arriva a d entro $\theta\}$.

Nel caso in cui $j=d$, si pone per convenzione che il cammino q sia il cammino vuoto.

L'insieme $P_{d\theta}$ può essere rappresentato tramite gli insiemi $ACT_d(e)$, $e \in E$; infatti:

$$P_{d\theta} = \bigcup_{e \in E} ACT_d(e).$$

La procedura **Build** esplora il grafo (V,E) a partire dalla destinazione d; ne visita la stella entrante alla ricerca degli archi $e = (u, t_u, d, t_d)$ utili per u verso d; il cammino costituito dal solo arco e è l'unico cammino costituente $ACT_d(e)$. I nodi u via via incontrati in questa scansione possono essere tanto nodi fermata quanto centroidi; nel primo caso l'arco $e \in V$ viene inserito in un insieme H degli archi candidati, da cui verrà successivamente estratto per essere analizzato; nel caso invece in cui $u \in C \setminus \{d\}$ tale arco non viene inserito nell'insieme H.

La scansione del grafo procede estraendo iterativamente dall'insieme H l'arco $e = (u, t_u, v, t_v)$ il cui tempo di partenza t_u da u è il tempo di partenza massimo tra quelli di tutti gli archi correntemente presenti in H. Si esplora la stella entrante di u, $BS(u)$, e per ogni arco $\overline{e} = (k, t_k, u, \overline{t}_u) \in BS(u)$ tale che $\overline{t}_u \leq t_u$, cioè che il tempo di arrivo a u di $\overline{e}$, $\overline{t}_u$, sia minore o uguale del tempo di ripartenza da u di e, t_u, si effettuano gli agganci tra l'arco $\overline{e}$ e tutti i cammini che partono da u con l'arco e, ossia i cammini di $ACT_d(e)$. Tali cammini vengono aggiunti all'insieme $ACT_d(\overline{e})$. Se l'arco $\overline{e}$ non è ancora presente in H, vi viene inserito per poter continuare a realizzare gli agganci e viene aggiunto all'insieme degli archi utili per k verso d, $U_d(k)$.

Si osservi che solo nel momento in cui l'arco $\overline{e}$ viene concatenato ad un dato cammino q di $ACT_d(e)$, è possibile controllare che il nuovo cammino generato $p = \{\overline{e}\} \cup q$ sia ammissibile in base alla definizione 3.1. La verifica dell'ammissibilità del cammino $p = \{\overline{e}\} \cup q$ viene eseguito dalla procedura **Feasible**, che restituisce **true** nel caso in cui p sia ammissibile, **false** altrimenti.

Contestualmente alla generazione di ogni nuovo cammino p, la procedura **Costo** ne calcola il costo generalizzato $c(p)$ in accordo alla (2.1).

La generazione dei cammini ha termine quando l'insieme H non contiene più archi da estrarre. Si noti che la selezione da H degli archi ordinata per tempi non crescenti di partenza permette la creazione in ordine temporale di tutti e soli i cammini ammissibili per d e θ. Infatti il generico arco e può essere estratto da H solo nel momento in cui è già stato generato tutto l'insieme $ACT_d(e)$ dei cammini attivi dell'arco e verso d.

La procedura **Build** può essere così specificata:

```
Procedure Build(P_dθ):
    begin
        for e:= 1 to m do ACT_d(e) := ∅;
        for i := 1 to n  do U_d(i) := ∅;
        foreach e = (u,t_u,d,t_d) ∈ BS(d) suchthat t_d≤ θ do
            begin
                p := {e};
                c(p) := Costo(p);
                ACT_d(e) := ACT_d(e) ∪ {p};
                U_d(u) := U_d(u)∪ {e};
                H := H ∪ {e}
            end;
        repeat
            select e = (u, t_u, v, t_v) from H suchthat t_u = max{t_z : (z, t_z, w, t_w) ∈ H}; H := H \ {e};
            foreach  ē = (k, t_k, u, t̄_u) ∈ BS(u) suchtath t̄_u ≤ t_u do
                foreach q ∈ ACT_d(e) do
                    if Feasible(ē, q)
                        then
                            begin
                                p := {ē} ∪ q;
                                c(p) := Costo(p);
                                ACT_d(ē) = ACT_d(ē) ∪ {p};
                                U_d(k) := U_d(k) ∪ {ē};
                                H = H ∪ {ē}
                            end
        until H = ∅
    end.
```

3.1.2 La generazione di P(max_ε)

Per ogni destinazione $d \in C$ e per ogni gruppo $k = (o, d, \tau)$ che abbia d come destinazione, la procedura **PathGen** seleziona da $P_{d\theta}$ l'insieme $P_{od\tau}$ di tutti i cammini che partono da o e arrivano a d entro τ; inoltre calcola il costo $\gamma_k = \gamma_{od\tau}$ del cammino ideale per il gruppo k:

$$\gamma_{od\tau} = \min \{c(p): p \in P_{od\tau}\}.$$

La procedura **PathGen** si avvale della procedura **Build** per la generazione dell'insieme $P(max_\varepsilon)$ di tutti i cammini ammissibili max_ε-equivalenti.

Si osservi che per selezionare $P_{od\tau}$ da $P_{d\theta}$ è sufficiente scandire l'insieme $U_d(o)$ degli archi utili per o verso d, e i corrispondenti insiemi attivi, non considerando quei cammini che arrivano a d oltre il tempo limite τ. La selezione di $P_{od\tau}$ da $P_{d\theta}$ e il calcolo di γ_k vengono effettuati dalla procedura **CalcCostId** che restituisce entrambi in uscita.

Per ogni cammino $p \in P_{od\tau}$ la procedura **PathGen** calcola il costo relativo $c_k(p)$, verifica la condizione di max_ε-equivalenza e, in caso di esito positivo, aggiunge il cammino p all'insieme $P(max_\varepsilon)$.

Poiché inoltre i valori distinti di costo relativo minori o uguali a max_ε rappresentano tutti i possibili valori di soglia effettivi, contestualmente all'aggiornamento di $P(max_\varepsilon)$ essa può costruire l'insieme $EPS' = \{\varepsilon_0,\ldots, \varepsilon_l\} \subseteq EPS$, $\varepsilon_0 = \varepsilon_{min}$ e $\varepsilon_l = max_\varepsilon$; EPS' viene ordinato in senso crescente dalla procedura **Sort**.

Schematicamente, la procedura **PathGen** può essere così formulata:

```
Procedure PathGen(max_ε):
    begin
        P(max_ε) := ∅; EPS' := ∅;
        foreach d∈ C do
            begin
                Build(P_dθ);
                foreach (o, d, τ) do
                    begin
                        CalcCostId(γ_odτ, P_odτ);
                        foreach p ∈ P_odτ do
                            begin
                                c_k(p) := c(p)/γ_odτ;
                                if c_k(p) ≤ max_ε
                                    then begin  P(max_ε) := P(max_ε) ∪ {p};
                                                EPS' := EPS' ∪ {c_k(p)} end
                            end
                    end
            end;
        Sort(EPS')
    end.
```

3.2 L'assegnazione ε*-minimale*

Grazie alla creazione dell'insieme $EPS' = \{\varepsilon_0,\ldots, \varepsilon_l\}$, la determinazione del valore di soglia minimale $\overline{\varepsilon}$ può essere realizzata attraverso la selezione dell'indice $r \in I = \{1,\ldots, l\}$ tale che $F(\varepsilon_r) \neq \emptyset$ e $F(\varepsilon_{r-1}) = \emptyset$; dunque $\overline{\varepsilon} = \varepsilon_r$.

Il problema di assegnazione ε-minimale può essere schematicamente risolto tramite la seguente procedura **Minimal**:

```
Procedure Minimal(ε̄, h):
    begin
        PathGen(max_ε);
        find r ∈ I suchthat F(ε_r) ≠ ∅ and F(ε_r-1) = ∅;
        ε̄ = ε_r; Select(P(ε̄)); h := Assign(P(ε̄))
    end.
```

in cui la procedura **Select** seleziona da $P(max_\varepsilon)$ l'insieme $P(\overline{\varepsilon})$ dei cammini $\overline{\varepsilon}$-equivalenti e la procedura **Assign** fornisce in uscita un flusso h $\overline{\varepsilon}$-ammissibile, ossia un flusso ε-minimale. Per la determinazione del flusso h la procedura **Assign** richiama un codice di programmazione lineare.

A seconda della tecnica utilizzata per la scansione dell'insieme I nella ricerca dell'indice r di soglia minimale, si possono ottenere differenti versioni della procedura **Minimal**. Nel seguito presentiamo tre classici approcci di scansione, quello incrementale, quello decrementale, e quello dicotomico.

3.2.1 L'approccio incrementale

L'approccio incrementale, a partire dall'indice minimo i=0, che corrisponde a ε_0=1, e per il quale sicuramente $F(\varepsilon_0) = \emptyset$, incrementa iterativamente tale indice fino a trovare il primo valore r per cui $F(\varepsilon_r) \neq \emptyset$. Alla generica iterazione i la procedura **Select** seleziona da $P(max_\varepsilon)$ l'insieme $P(\varepsilon_i)$ dei cammini ε_i-equivalenti e la procedura **Assign** calcola in h un flusso ε_i-ammissibile, se un tale flusso esiste, altrimenti restituisce $h = \mathbf{0}$. Il procedimento si arresta alla prima iterazione (la (r-1)-esima) per cui $h \neq \mathbf{0}$. La procedura **Minimal** può essere così specificata:

```
Procedure MinimalInc(ε̄, h):
    begin
        PathGen(max_ε);
        i:= 0; h := 0;
        repeat
            i := i+1; Select(P(ε_i)); h := Assign(P(ε_i))
        until h ≠ 0;
        ε̄ = ε_i
    end.
```

È evidente che l'efficienza computazionale della procedura **MinimalInc** dipende strettamente dalla tecnica di selezione dei cammini ε_i-equivalenti dall'insieme $P(max_\varepsilon)$ impiegata nella procedura **Select**; poiché tale aspetto è comune a tutte e tre le versioni della procedura **Minimal**, ne rimandiamo la trattazione a un successivo paragrafo.

3.2.2 L'approccio decrementale

In prima istanza si potrebbe pensare a un approccio decrementale perfettamente simmetrico a quello incrementale, che partendo dall'indice massimo $i = l$, corrispondente a $\varepsilon_l = max_\varepsilon$ e per il quale sicuramente $F(\varepsilon_l) \neq \emptyset$, decrementi iterativamente tale indice fino a trovare il primo valore i tale che $F(\varepsilon_i) = \emptyset$, e restituisca $\bar{\varepsilon} = \varepsilon_{i+1}$ come valore di soglia minimale.

In conseguenza delle seguenti osservazioni l'approccio decrementale può essere realizzato in modo efficiente. Consideriamo la generica iterazione $i \geq r$ e sia $hnew$ il flusso fornito dalla procedura **Assign**; $hnew$ viene determinato su un insieme *ridondante* di cammini, $P(\varepsilon_i)$, nel senso che $P(\varepsilon_i)$ può contenere più cammini di quelli che sono strettamente necessari a garantire l'esistenza di un flusso ammissibile (i cammini di $P(\varepsilon_r)$), con la conseguenza che il flusso $hnew$ potrebbe avere componenti strettamente positive su una parte soltanto dei cammini di $P(\varepsilon_i)$. Pertanto, dopo aver calcolato il flusso $hnew$ tramite la procedura **Assign**, si determina l'indice del *valore di soglia effettivamente utilizzato* nella soluzione $hnew$ corrente, come l'indice j tale che

$$\varepsilon_j = \max \{c_k(p) : hnew_p > 0, p \in P_k(\varepsilon_i), k = 1,\ldots, K\}, \qquad (3.1)$$

si effettua la successiva iterazione sull'indice j-1, ossia si verifica se esiste ancora un flusso ammissibile su un insieme di cammini, $P(\varepsilon_{j-1})$, che non contiene più tutti i cammini realmente utilizzati in $hnew$.

Se un tale flusso esiste, si reitera il procedimento, altrimenti l'algoritmo termina con j indice di soglia minimale ($j = r$). La procedura **MinimalDec**, in cui la procedura **Ind** calcola l'indice di soglia effettiva in accordo alla (3.1), può essere schematizzata come segue:

```
Procedure MinimalDec(ε̄, h):
    begin
        PathGen(max_ε);
        hnew:= Assign(P(max_ε));
        while hnew ≠ 0 do
            begin
                h := hnew; j := Ind(hnew);
                Select(P(ε_{j-1})); hnew:= Assign(P(ε_{j-1}))
            end;
        ε̄ = ε_j
    end.
```

L'idea di sfruttare le caratteristiche di costo dei cammini di volta in volta utilizzati nella soluzione corrente può essere ulteriormente motivata e rafforzata da una scelta opportuna della funzione obiettivo per il problema risolto dalla procedura **Assign**. Utilizzando infatti la seguente funzione obiettivo:

$$\min \sum_{k=1}^{K} \sum_{p \in P_k(\varepsilon)} c_k(p) h_p$$

ossia la somma dei flussi di cammino moltiplicati per i costi relativi, ad ogni iterazione il codice di P.L. utilizzato nella procedura **Assign** tende per costruzione a assegnare il flusso sui cammini di costo relativo minimo. Ciò porta a ritenere che, in media, il numero di iterazioni della procedura **MinimalDec** così modificata possa essere significativamente inferiore a quello della versione precedente, come peraltro hanno confermato i primi risultati sperimentali.

3.2.3 L'approccio dicotomico

Nel terzo approccio si effettua una ricerca dicotomica sull'insieme degli indici $I = \{1,\ldots, l\}$, con la sola particolarità che, nel caso in cui esista un flusso *hnew* ε_i-ammissibile rispetto all'indice intermedio i di volta in volta considerato, per determinare l'estremo superiore del nuovo intervallo di variabilità degli indici si impiega la procedura **Ind** in modo tale da sfruttare, anche in questo approccio, le caratteristiche di costo dei cammini effettivamente utilizzati nella soluzione corrente. L'algoritmo **Minimal** si traduce nella seguente procedura **MinimalDic**:

```
Procedure MinimalDic(ε̄, h):
    begin
        PathGen(max_ε);
        first := 0; last := l;
        repeat
            i := ⌊(first+last) / 2⌋;
            Select(P(ε_i)); hnew:= Assign(P(ε_i));
            if hnew = 0   then first := i
                          else begin last := Ind(hnew); h := hnew end
        until first ≥ (last-1);
        ε̄ = ε_last
    end.
```

3.2.4 La selezione dei cammini

Indipendentemente dall'approccio algoritmico scelto per la realizzazione della procedura **Minimal**, la tecnica utilizzata dalla procedura **Select** per la selezione dei cammini ε_i-equivalenti dall'insieme $P(max_\varepsilon)$ influisce sull'efficienza dell'algortimo stesso. Dal momento che $P(\varepsilon_i) \subset P(\varepsilon_j)$, $\forall\ i, j \in I$ tali che $i < j$, non è assolutamente necessario che la procedura **Select** a ogni iterazione selezioni ex novo da $P(max_\varepsilon)$ tutti i cammini attivi per il corrente valore di soglia. Infatti, per quanto riguarda l'approccio incrementale, alla generica iterazione i è sufficiente selezionare da $P(max_\varepsilon)$ solo quei cammini che non erano attivi per ε_{i-1}, ma che lo sono rispetto a ε_i, ed unirli a $P(\varepsilon_{i-1})$ per ottenere $P(\varepsilon_i)$. Nell'approccio decrementale, invece, basta eliminare dall'insieme corrente di cammini, $P(\varepsilon_i)$, tutti quelli il cui costo relativo è maggiore o uguale al valore di soglia effettiva ε_j, con j determinato in base alla (3.1), per

ottenere $P(\varepsilon_{j-1})$. Entrambe le considerazioni valgono ovviamente per l'approccio dicotomico.

Questa osservazione è di particolare rilevanza non solo per quanto concerne la procedura **Select**, ma anche per la stretta connessione che intercorre tra l'aggiornamento dell'insieme $P(\varepsilon_i)$ e l'implementazione della procedura **Assign**. Risolvendo infatti il problema della determinazione di un eventuale flusso ε_i-ammissibile tramite un codice di programmazione lineare che permetta la riottimizzazione, i cammini da aggiungere o eliminare da $P(\varepsilon_i)$ vengono tradotti in colonne rispettivamente da aggiungere o eliminare dalla matrice dei vincoli del problema di P.L.; sulla matrice così modificata si può eseguire la riottimizzazione con la conseguenza di snellire notevolmente il carico computazionale complessivo.

4. Un esempio e alcuni aspetti implementativi

Allo stato attuale sono state implementate soltanto le procedure **MinimalInc** e **MinimalDec** in quanto i risultati confortanti ottenuti dalle prove effettuate con la procedura **MinimalDec** ci hanno indirizzato a proseguire il nostro lavoro più verso un confronto accurato tra questi due tipi di approccio che verso l'implementazione dell'approccio dicotomico.

Le due procedure sono state implementate in linguaggio C ANSI standard, e provate su una workstation HP9000/712-80 con sistema opertivo HP-UX 9.05.

Nel seguito descriviamo una rete test, la domanda di trasporto, il flusso ε-minimale restituito e i tempi di elaborazione delle varie fasi del codice.

4.1 *La rete test*

La rete test ha 13 nodi fermata, 7 nodi centroide, 20 archi di linea e 13 archi a piedi bidirezionali che connettono nodi centroide e nodi fermata; essa è schematizzata nella Figura 4.1.

Tabella 4.1: Le corse

n. linea	**n. corsa**	**percorso**						
1	1	(1; 06:30)	(2; 06:45)	(3; 06:55)	(6; 07:07)	(9; 07:22)	(12; 07:32)	(13; 07:55)
1	2	(1; 07:00)	(2; 07:15)	(3; 07:25)	(6; 07:37)	(9; 07:52)	(12; 08:02)	(13; 08:25)
2	1	(4; 06:30)	(3; 06:43)	(6; 06:55)	(9; 07:10)			
2	2	(4; 07:00)	(3; 07:13)	(6; 07:25)	(9; 07:40)			
3	1	(13; 06:30)	(11; 06:40)	(8; 06:55)	(9; 07:05)	(10; 07:20)		
3	2	(13; 07:00)	(11; 07:10)	(8; 07:25)	(9; 07:35)	(10; 07:50)		
4	1	(12; 06:40)	(11; 06:47)	(8; 07:02)	(5; 07:17)	(2; 07:27)		
4	2	(12; 07:10)	(11; 07:17)	(8; 07:32)	(5; 07:47)	(2; 07:57)		
5	1	(5; 06:50)	(6; 07:00)	(7; 07:15)	(10; 07:30)			
5	2	(5; 07:20)	(6; 07:30)	(7; 07:45)	(10; 08:00)			

Il servizio offerto è costituito da 5 linee distinte, per ognuna delle quali sono istituite 2 corse; per semplicità abbiamo supposto che non vi siano *tempi di attesa a bordo*, ossia che tutti gli autobus ripartano da ogni fermata nello stesso istante in cui vi giungono. La tabella 4.1 riporta sulle righe le corse, e sulle colonne, rispettivamente, il numero di linea, il numero di

corsa, e il percorso della corsa stessa, descritto come sequenza di coppie (i, t_i), in cui i è il nodo fermata attraversato, e t_i l'orario di partenza della corsa dal nodo i; l'ultima coppia della sequenza indica rispettivamente il nodo fermata terminale della corsa e l'orario di arrivo a esso.

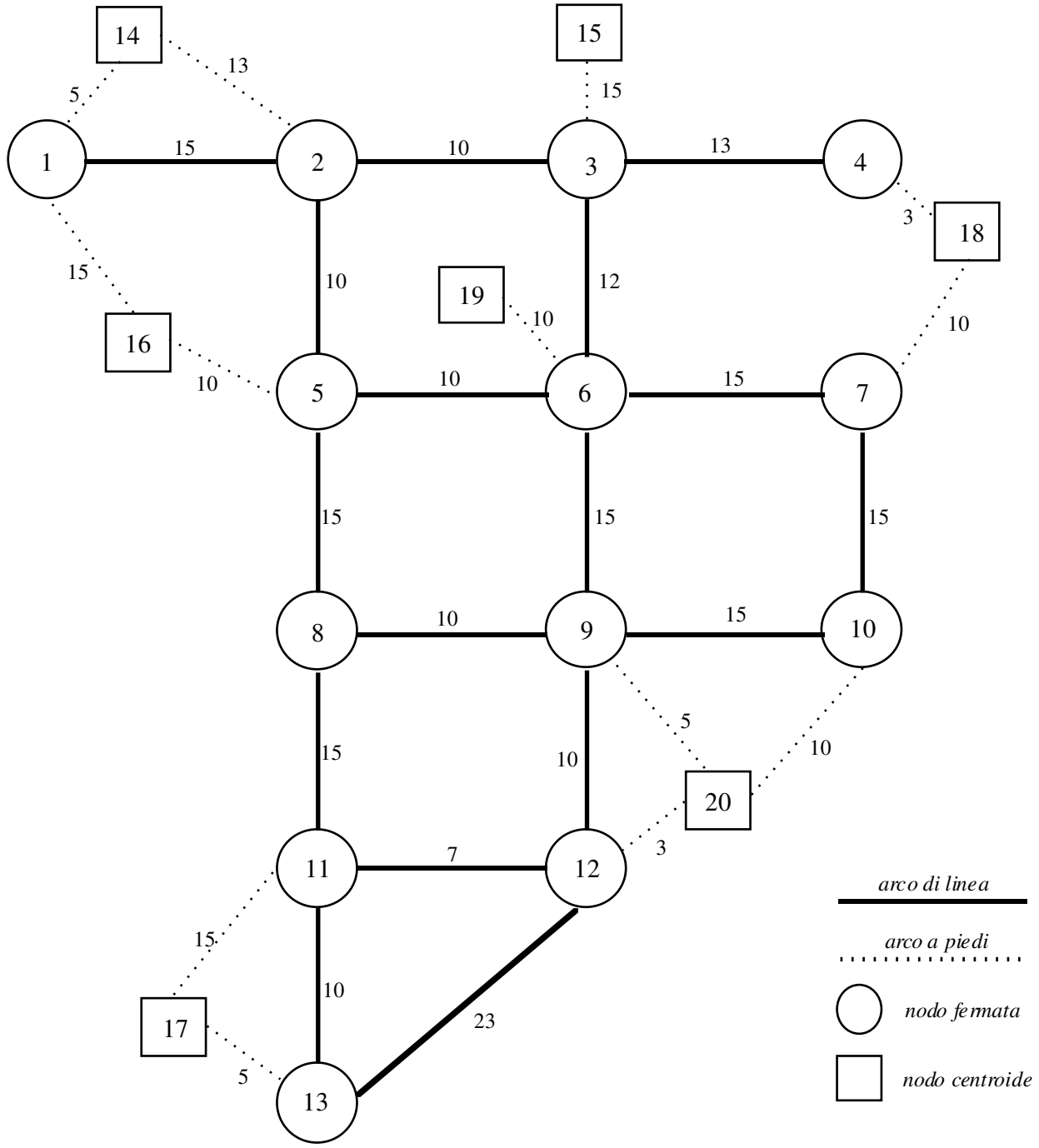

Figura 4.1: La topologia della rete con i tempi di percorrenza

La capacità è stata fissata in 55 posti a sedere per tutti gli autobus utilizzati.

Ai coefficienti α, β, δ, γ, η della funzione costo generalizzato data in (2.1) sono stati assegnati i seguenti valori: $\alpha = 1$, $\beta = 3$, $\gamma = 10$, $\delta = 5$, $\eta = 1$.

4.2 *La domanda di trasporto*

La domanda è suddivisa in 35 gruppi di utenti. Sono stati presi in considerazione due possibili tempi limite all'arrivo: le 7:40 e le 8:40. Riportiamo rispettivamente nelle tabelle 4.2 e 4.3 le due matrici domanda relative, $D_{7:40}$ e $D_{8:40}$, indicanti per ogni coppia di nodi centroide (i, j), $i, j \in C$, il numero di passeggeri che intende spostarsi dal centroide i al centroide j entro il corrispondente tempo limite.

Tabella 4.2

$D_{7:40}$	**14**	**15**	**16**	**17**	**18**	**19**	**20**
14	0	10	0	0	0	0	46
15	0	0	0	0	0	15	25
16	0	18	0	0	36	0	4
17	15	0	10	0	0	0	50
18	0	25	0	0	0	10	50
19	0	0	0	0	10	0	25
20	15	0	15	25	0	0	0

Tabella 4.3

$D_{8:40}$	**14**	**15**	**16**	**17**	**18**	**19**	**20**
14	0	0	0	11	0	0	11
15	0	0	0	9	0	0	15
16	0	0	0	5	31	0	4
17	15	0	10	0	0	0	50
18	0	0	0	15	0	0	50
19	0	0	0	10	10	0	25
20	10	0	15	0	0	0	0

4.3 *La soluzione fornita*

La rete test è stata progettata per verificare la robustezza del codice in presenza di situazioni di congestione indotte dalla presenza dei vincoli di capacità sulle corse. Si consideri ad esempio l'arco della corsa 1 della linea 1 che parte alle 7:07 dal nodo 6 ed arriva al nodo 9 alle 7:22; si può facilmente verificare che tale arco appartiene a cammini ideali dei gruppi (14, 20, 7:40), (15, 20, 7:40), (16, 20, 7:40), (19, 20, 7:40) e, poiché la domanda complessiva per i suddetti gruppi è 100 mentre la capacità dell'arco è 55, su tale arco si verifica congestione; inoltre non esiste una assegnazione di flusso in cui tutti i gruppi percorrano cammini ideali.

Il valore di max_ε fornito in ingresso, $max_\varepsilon = 5$, è tale da generare tutti i cammini ammissibili ($max_\varepsilon > \varepsilon_{max}$). Le due procedure hanno fornito come intervallo di variabilità di ε l'intervallo [1, 4.3], e l'insieme $EPS = EPS'$ dei valori possibili di soglia è risultato di 114 elementi.

La procedura **MinimalInc** ha restituito $\bar{\varepsilon} = \varepsilon_{52} = 2.06$, ovviamente dopo 51 riottimizzazioni; il corrispondente flusso ε-minimale è visualizzato nella figura 4.2: a ogni arco di linea è associata la coppia (f_1, f_2) di valori di flusso relativi rispettivamente alla prima e alla seconda corsa della linea stessa.

La procedura **MinimalDec** ha fornito, dopo una sola riottimizzazione, una soluzione identica a quella della procedura **MinimalInc** salvo per il carico di flusso sull'arco dal nodo fermata 9 al nodo fermata 12, relativo alla seconda corsa della linea 1: tale flusso è di 50 unità anziché 55, ma ciò è motivato dal fatto che per gli utenti di tale corsa diretti al nodo centroide 20 scendere al nodo 9 o al nodo 12 è del tutto equivalente in termini di costo generalizzato.

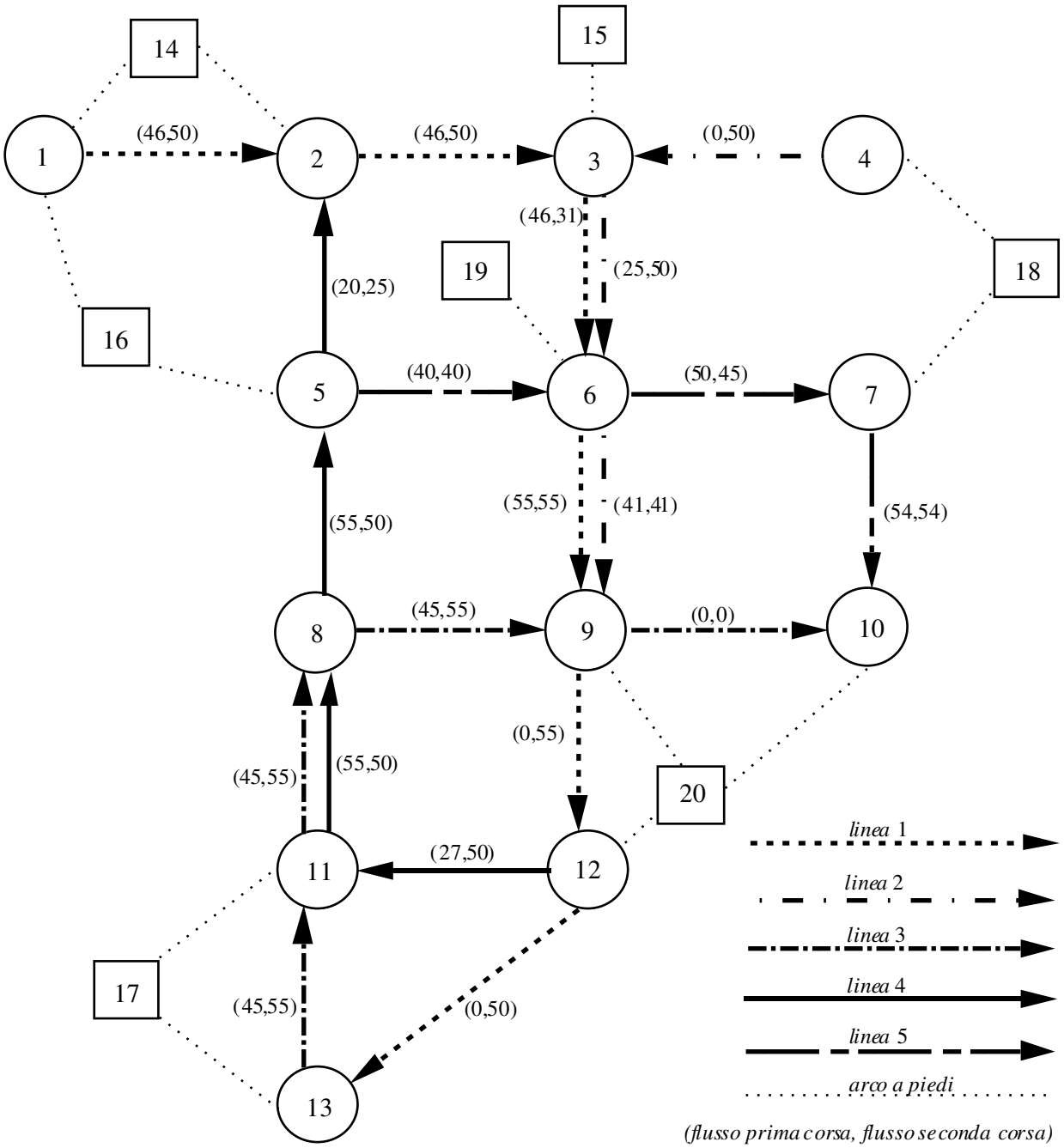

Figura 4.2: Il flusso fornito dalla procedura **MinimalInc**

Le due procedure sono state provate anche variando la capacità di alcune corse e/o la domanda per qualche gruppo di utenti: tanto i valori della soglia minimale quanto i flussi ottenuti risultati uguali, a meno di variazioni dello stesso tipo di quello appena esposto, per entrambi gli approcci.

4.4 *I tempi di elaborazione*

Per entrambe le procedure **MinimalInc** e **MinimalDec** sono stati calcolati i diversi tempi di CPU:

- il tempo t_g di esecuzione della procedura **PathGen**, che ha generato 453 tra cammini e tratti di cammino, di cui 259 identificati come *max*_ε equivalenti e scaricati su memoria scondaria;
- il tempo t_o della prima ottimizzazione;

- il tempo t_r complessivo di riottimizzazione;
- il tempo di elaborazione totale t_{tot}.

I tempi, espressi in centesimi di secondo, sono la media dei tempi ottenuti per 20 differenti esecuzioni e sono stati calcolati al netto dei tempi di accesso alla memoria secondaria.

Le tabelle 4.4 e 4.5 sintetizzano il numero n_r di riottimizzazioni e i tempi t_g, t_o, t_r, t_{tot}, minimi, medi e massimi ottenuti sulle 20 esecuzioni rispettivamente dalla procedura **MinimalInc** e dalla **MinimalDec**:

Tabella 4.4: I tempi di CPU per la procedura **MinimalInc**

	t_g	t_o	n_r	t_r	t_{tot}
min	2.00	1.00	51	30.00	33.00
med	4.75	2.00	51	38.25	45.00
max	7.00	4.00	51	44.00	50.00

Tabella 4.5: I tempi di CPU per la procedura **MinimalDec**

	t_g	t_o	n_r	t_r	t_{tot}
min	2.00	3.00	1	0.00	8.00
med	4.75	6.40	1	1.15	12.30
max	7.00	9.00	1	2.00	16.00

Come era lecito aspettarsi, nella procedura **MinimalDec** i tempi t_o di esecuzione della prima ottimizzazione sono decisamente più elevati, in quanto la matrice di P.L. iniziale di questa procedura codifica tutti i cammini *max*_ε-equivalenti, mentre quella della procedura **MinimalInc** è relativa solo ai cammini di costo relativo uguale a ε_1.

Per quanto riguarda il numero di riottimizzazioni, mentre la procedura **MinimalInc** ne effettua 51, la procedura **MinimalDec** ne effettua una sola; questo dipende in sostanza dal fatto che la procedura **MinimalDec** sfrutta a ogni iterazione le informazioni di costo sul flusso ε_i-ammissibile ottenuto alla iterazione precedente, come spiegato nel paragrafo 3.2.2. Sulla rete test una sola iterazione si è rivelata sufficiente a eliminare dalla matrice di P.L. tutti i cammini superflui rispetto alla $\bar{\varepsilon}$-ammissibilità. La differenza tra i numeri di riottimizzazioni nelle due procedure si ripercuote evidentemente sui tempi t_r e t_{tot}. In merito al tempo di riottimizzazione t_r, si osservi che nella procedura **MinimalInc** il tempo medio di riottimizzazione è 0.75 centesimi di secondo, mentre nella procedura **MinimalDec** si ha un tempo medio di 1.15 centesimi, in virtù delle differenti dimensioni della matrice del problema di P.L. risolto ad ogni iterazione.

Il tempo totale medio di esecuzione, dato dalla somma dei tempi medi t_g di generazione dei cammini, t_o di prima ottimizzazione e t_r di riottimizzazione, riflette in pieno le considerazioni fin qui riportate; in particolare, se si considerano tali tempi senza tener conto della fase di generazione dei cammini, si può evincere che mediamente il tempo complessivo di ricerca dell'ε minimale per la procedura **MinimalInc** è approssimativamente 5.3 volte quello necessario alla **MinimalDec**.

Concludendo, la stabilità dei tempi di esecuzione forniti sulla rete test e le considerazioni appena esposte inducono a ipotizzare che l'approccio decrementale, più che quello incrementale, possa costituire uno strumento particolarmente efficiente per la determinazione di un flusso ε-minimale nei casi in cui il valore di partenza *max*_ε sia sufficientemente limitato.

Bibliografia

[1] Alfa, A., "Departure rate and route assignment of commuter traffic during peak period" *Transportation Research* **23B** (1989) 337-344.

[2] Carraresi, P., F. Malucelli e S. Pallottino, "On the regional mass transit assignment problem", in (A. Sciomachen, ed.) *Optimization in Industry 3*, Wiley (1995) 19-33.

[3] Carraresi, P., F. Malucelli e S. Pallottino, "Regional mass transit assignment with resource constraints", Università di Pisa, Dipartimento di Informatica, TR-11/94, in corso di stampa su *Transportation Research* (1996).

[4] Cascetta, E., "Modelli di scelta dell'itinerario e del percorso su reti di trasporto", Seminario su Assegnazione alle Reti di Trasporto, Roma, Università di Tor Vergata (1995).

[5] Dellasette, M., "Analisi e valutazione di sistemi di trasporto pubblico con caratteristiche di tipo extraurbano, con particolare riferimento ai modelli di assegnazione e ripartizione modale", Tesi di Dottorato di Ricerca in Ingegneria dei Trasporti, Università di Trieste-Politecnico di Torino, Torino (1995).

[6] Florian, M., "An introduction to network models used in transportation planning", in (M. Florian, ed.) *Transportation Planning Models*, North-Holland, Amsterdam (1984) 137-152.

[7] Magnanti, T.L., "Models and algorithms for predicting urban traffic equilibria", in (M. Florian, ed.) *Transportation Planning Models*, North-Holland, Amsterdam (1984) 152-185.

[8] Mahmassani, H.S. e R. Herman, "Dynamic user equilibrium departure time and route choice on idealized traffic arterials" *Transportation Science* **18** (1984) 362-385.

[9] Nguyen, S., "A unified approach to equilibrium methods for traffic assignment", in (M. Florian, ed.) *Traffic equilibrium methods*, Springer-Verlag, Berlin (1976) 148-182.

[10] Nguyen, S. e S. Pallottino, "Equilibrium traffic assignment for large scale transit network" *European Journal of Operational Research* **37** (1988) 176-186.

[11] Nguyen, S., S. Pallottino e F. Malucelli, "A modelling framework for passenger assignment on transport network with time-tables", Université de Montréal, Centre de Recherche sur les Transports, publ. 94/47 (1994), sottoposto a *Transportation Science*.

[12] Sumi, T., Y. Matsumoto e Y. Miyaki, "Departure time and route choice of commuters on mass transit systems" *Transportation Research* **24B** (1990) 247-262.