UNIVERSITÁ DEGLI STUDI DI PADOVA

FACOLTÁ DI INGEGNERIA

DIPARTIMENTO DI TECNICA E GESTIONE DEI SISTEMI INDUSTRIALI

CORSO DI LAUREA IN INGEGNERIA GESTIONALE

Tesi di Laurea Magistrale

SUPPLY NETWORK ANALYSIS:

STUDIO DEI FLUSSI LOGISTICI DI RICIRCOLO

Relatore: Ch.ma Prof.ssa DARIA BATTINI

Correlatore: Ing. ANNA AZZI

Laureando: DARIO SCUDELLARO

ANNO ACCADEMICO 2010 - 2011

Indice

Sommario

Questo lavoro di tesi si propone di introdurre una nuova metodologia per l'analisi della struttura dei flussi logistici nelle reti di fornitura industriali, focalizzandosi in particolar modo sullo studio dei flussi di ricircolo che si verificano al loro interno.
Gli strumenti impiegati nel seguito della trattazione derivano dalla network analysis, approccio diffuso per lo studio degli ecosistemi naturali. Grazie all'analogia che esiste tra i network ecologici e quelli industriali è difatti possibile studiare i secondi sfruttando in modo innovativo concetti e tecniche applicati con successo in ambito ecologico.
In particolare, lo strumento per l'analisi dei cicli, opportunamente adattato, consente di stimare l'entità dei flussi generati mediante ricircoli.
Scopo del lavoro è dimostrare la fattibilità e la potenzialità di questa metodologia che consente di ottenere delle misure quantitative in grado di fotografare aspetti riguardanti la complessità, il livello di organizzazione, il fenomeno dei ricircoli e la sostenibilità di un supply network, da un punto di vista olistico.

Introduzione

Termini come riuso, riciclaggio, rilavorazione e gestione dei prodotti a fine vita utile (*end-of-life products*), sono entrati a far parte del comune lessico manageriale in quanto identificano delle attività di crescente rilevanza per le aziende (Pagell *et al.*, 2007). Tali attività rientrano per molteplici aspetti nella sfera di controllo della logistica inversa, il cui compito è quello di gestire i flussi di materiali grezzi, semilavorati o prodotti finiti, dal punto della supply chain in cui sono generati fino al centro in cui verranno riutilizzati (previo opportuno trattamento di recupero) oppure smaltiti.
Si instaurano quindi dei flussi logistici inversi che, dalla produzione, dalla distribuzione o dai punti vendita, vanno in direzione di qualche centro di recupero, di ridistribuzione o di smaltimento.
Nella realtà accade spesso che i flussi logistici inversi si fondono con quelli diretti in più punti della supply chain, formando la cosiddetta "supply chain a ciclo chiuso" (*Closed Loop Supply Chain*, CLSC).
La logistica inversa diventa quindi una pratica del supply chain management intesa come la gestione dei flussi fisici chiusi e funzionali (Ruggeri Laderchi e Payaro, 2004) che comportano il ricircolo di materiale, semilavorati, prodotti finiti, pallet, UDC etc.
Si rende quindi necessario uno studio di tali flussi logistici per arrivare almeno a stimare l'entità del fenomeno dei ricircoli all'interno di un supply network.
Da questa considerazione nasce il lavoro di tesi di seguito presentato, che si articola in tre capitoli e un'appendice.

In particolare, il capitolo 1 propone una classificazione dei flussi logistici di ricircolo che si riscontrano nella realtà industriale e che sono riconosciuti anche dalla letteratura scientifica nell'ambito degli studi inerenti alla logistica inversa.
Assumendo una prospettiva basata sui processi, che arriva a suddividere la logistica inversa in quattro parti (raccolta, ispezione/selezione/ordinamento, trattamento e redistribuzione dei prodotti), è possibile individuare diverse tipologie di ricircolo a seconda del trattamento subito da un prodotto: esiste uno spettro di possibili scelte da intraprendere, che vanno dal riuso diretto allo smaltimento in discarica, passando per le opzioni di recupero.
Viene inoltre individuata un'altra tipologia di ricircolo, qui definita come "conto-lavoro" che, seppur riconosciuta dalla letteratura, non viene contemplata all'interno degli studi sulla logistica inversa.

Il capitolo 2 introduce alla network analysis, pratica diffusa soprattutto nell'ambito ecologico per lo studio degli ecosistemi naturali: si tratta di una serie di strumenti applicativi, metodologie e procedure che, opportunamente adattati, bene si prestano a descrivere la struttura di una rete di fornitura industriale.
In particolare, vengono presentati gli strumenti per la mappatura del network e per il calcolo degli indici di performance (indici entropici) e dell'indice di Finn, i quali consentono, rispettivamente, di misurare la complessità (ovvero il livello di organizzazione) del network e di fornire una stima della percentuale dei flussi generati da cicli.
Infine si approfondisce lo studio sui ricircoli nelle reti logistiche, proponendo un metodo di analisi basato sulle categorie di ricircolo identificate nel primo capitolo, in modo da ottenere un quadro più esaustivo sul fenomeno.

Nel capitolo 3 viene applicata la supply network analysis su delle istanze test modellate grazie a quanto ricavato dalla letteratura scientifica: nello specifico, le strutture dei flussi logistici dei network istanza test, si rifanno a quelle descritte nel lavoro di Blackburn *et al.* (2004).
L'intera analisi viene supportata da un software sviluppato ad hoc in MATLAB che implementa tutti gli step della metodologia, introdotti nel secondo capitolo: dalla mappatura del network al calcolo degli indici entropici, fino all'analisi dei ricircoli.
Quindi si confronteranno tra loro i risultati di ogni istanza test per vedere il comportamento degli indici presentati sempre nel secondo capitolo, con particolare attenzione all'indice di ricircolo di Finn.

In appendice si riporta il codice MATLAB del software sviluppato per la supply network analysis.

CAPITOLO 1

Flussi di ricircolo nei network

1.1.Logistica Inversa

Considerata in passato dalle organizzazioni come il processo mediante il quale i beni di consumo potevano essere restituiti nel caso si verificasse un errore nella consegna (un articolo difettoso, piuttosto che un ordine non corretto), un maggior numero di attività sono oggigiorno entrate a far parte della logistica inversa (Pagell *et al.*, 2007; Simpson, 2010).

Negli ultimi 20-30 anni, dall'ultimo anello della supply chain tradizionale[1], è emersa una gamma completamente nuova di beni. Essa include (Blumberg, 2005):

- prodotti guasti, difettosi o funzionanti e restituiti ai *retailer*, ma che possono essere riparati o direttamente riutilizzati;
- prodotti obsoleti, ma che possiedono ancora valore;
- prodotti invenduti dai *retailer*;
- prodotti richiamati (*recalled products*);
- parti riparate nel campo (ad es. durante l'assistenza fornita al cliente) e che non sono ancora alla fine della loro vita utile;
- beni che, dopo aver svolto il loro uso originario, possono avere un secondo utilizzo;
- rifiuti (speciali) che devono essere contabilizzati e smaltiti;
- contenitori riutilizzabili.

Guide e Van Wassenhove (2002) definiscono la logistica inversa come "la serie di attività con lo scopo di rimuovere, smaltire e ridistribuire nei mercati i materiali in eccesso, come imballaggi, merce usata, scarti di produzione e prodotti difettosi, richiamati dall'azienda o restituiti perché in garanzia".

Si riscontra un aumento delle attività legate alla logistica inversa, determinato da molteplici cause (Blumberg, 2005), tra cui:

[1] La supply chain tradizionale, gestita dalla logistica diretta, individua la parte di rete di fornitura che generalmente parte dai fornitori, passa per il produttore originale (*Original Equipment Manufacturer*, OEM), quindi per i grossisti e termina con i clienti finali.

- le tematiche ambientali regolamentate dalla legislazione o comunque osservate dai consumatori, sempre più attenti e sensibili a tali argomenti. Il produttore originale, soprattutto se opera nel mercato dell'elettronica di consumo, è spesso responsabile per legge del recupero e dello smaltimento del prodotto a fine vita utile;
- l'aumento del numero di beni restituiti dai clienti a fronte delle politiche di restituzione del prodotto adottate dalle imprese (tattiche di vendita). Solitamente, le grandi catene di distribuzione prendono accordi con i fornitori per restituire loro la merce: mentre in passato tale ritorno era riservato esclusivamente ai prodotti difettosi, oggi coinvolge anche quelli in perfetta condizione che non sono stati venduti o che un cliente ha restituito ad esempio perché non è soddisfatto dell'acquisto;
- l'accorciamento del ciclo di vita dei prodotti. I prodotti diventano obsoleti sempre più rapidamente e ciò potrebbe riflettersi in un maggior flusso di ritorno;
- la spinta a ridurre i costi. Le imprese cercano di recuperare valore dai prodotti e dai componenti potenzialmente ancora in buono stato, attraverso il loro riutilizzo o il riciclaggio di materiale. Per questo alcune aziende ritirano i beni che possiedono ancora valore, non più riconosciuto dal consumatore che, ad esempio, decide di sostituire un elettrodomestico non ancora arrivato a fine vita utile;
- la diffusione della pratica dell'e-commerce. L'incremento considerevole delle vendite effettuate via Internet è proporzionale all'aumento del flusso di ritorno per il fatto che i consumatori, acquistando merce "a scatola chiusa", spesso rimangono delusi o insoddisfatti del proprio acquisto;
- l'aumento di domanda per riparazioni, rilavorazioni (*remanufacturing*[2]), aggiornamenti o ritarature;
- il maggiore utilizzo di contenitori a rendere e/o riutilizzabili;
- la gestione dei prodotti in garanzia. Spesso gli articoli restituiti dai clienti ai *retailer* vengono fatti tornare nella sede principale per testarli e determinare la loro disposizione;
- la proliferazione di attività di noleggio. Ne consegue il ritorno di merci, attrezzature e apparecchi che sono stati usati;
- i richiami di prodotto. Il produttore può decidere di ritirare i prodotti dal mercato se ad esempio riscontra delle potenziali cause di guasto che potrebbero mettere a repentaglio la sicurezza dell'utilizzatore.

---

[2] Possibili traduzioni del termine *remanufacturing* possono essere: rilavorazione, riproduzione rifabbricazione, rigenerazione. Qui si è scelto di usare il termine "rilavorazione".

Generalmente l'OEM si trova nella posizione migliore all'interno del supply network per poter controllare i flussi di ritorno (Blumberg, 2005).
Quando un'impresa gestisce l'intero processo di fornitura sia verso il cliente finale che quello di ritorno, si ottiene quello che in letteratura viene chiamato "catena di fornitura a ciclo chiuso" (*Closed Loop Supply Chain*, CLSC).

1.2.CLSC: prospettiva basata sui processi

Guide e Van Wassenhove (2009) definiscono la gestione di una supply chain a ciclo chiuso come "la progettazione, il controllo e la gestione di un sistema per massimizzare la creazione di valore durante tutto il ciclo di vita di un prodotto attraverso il recupero dinamico di valore dai diversi tipi e volumi di flussi di ritorno nel tempo".
Le supply chain a ciclo chiuso variano da quelle che presentano un ritorno di prodotti per ragioni commerciali (*commercial returns*[3]) che quindi vengono ridistribuiti ai consumatori dopo veloci trattamenti, a quelle in cui vengono raccolti i prodotti giunti a fine vita utile, quindi riportati indietro lungo la catena di fornitura per disassemblarli, rilavorare le parti (*remanufacturing*) e riciclare i materiali (Van Nunen e Zuidwijk, 2004).
Le catene di fornitura a ciclo chiuso possono essere esaminate da tre diverse prospettive: dei consumatori, dei prodotti e dei processi (Dekker *et al.*, 2004).
La prima prospettiva è quella relativa alle informazioni raccolte dal CRM (*Customer Relationship Management*): disporre di questi dati consente di comprendere quali sono i servizi rilevanti per il cliente e come si possono migliorare.
La prospettiva basata sui prodotti si riferisce al recupero di informazioni sullo stato del prodotto e delle sue parti: monitorarle permette di anticipare il ritorno dei prodotti e di ottimizzare il recupero di valore dal prodotto stesso.
La prospettiva dei processi, invece, consente di individuare le diverse opzioni di recupero dei prodotti (*product recovery*): adottare tale punto di vista porta a riconoscere quali sono le attività di maggior rilevanza nella logistica inversa e dove esiste un margine di riduzione dei costi.
In linea con l'obiettivo di questo primo capitolo di tesi di individuare le tipologie di ricircoli all'interno delle supply chain, si assume, nel seguito della trattazione, l'ultima prospettiva presentata.

---

[3] Negli USA, sono i prodotti liberamente restituiti dai consumatori ai rivenditori entro 30 o 90 giorni dall'acquisto (Blackburn *et al.*, 2004), a seguito di politiche commerciali del tipo "soddisfatti o rimborsati".

La prospettiva basata sui processi, infatti, segmenta le attività della logistica inversa in quattro step (Dekker *et al*., 2004):

1. raccolta dei beni (ad esempio mediante punti di raccolta situati presso i punti di vendita esistenti);
2. ispezione/selezione/ordinamento dei prodotti raccolti;
3. trattamento dei prodotti attraverso un'opzione di recupero;
4. redistribuzione dei beni nel mercato primario o secondario.

Thierry *et al.* (1995) propongono un modello di catena di fornitura "integrata" (figura 1.1): si distinguono chiaramente i flussi ciclici che formano un anello chiuso in corrispondenza dei punti della supply chain in cui logistica diretta ed inversa si fondono. Secondo lo schema, i prodotti e i componenti di ritorno dai clienti finali possono essere riusati/rivenduti direttamente o recuperati o smaltiti (mediante incenerimento o interramento in discariche).

Thierry *et al*. individuano cinque possibili alternative nella gestione del recupero dei prodotti: riparazione (*repair*), revisione (*refurbishing*[4]), rilavorazione (*remanufacturing*), cannibalizzazione (*cannibalization*) e riciclaggio (*recycling*).

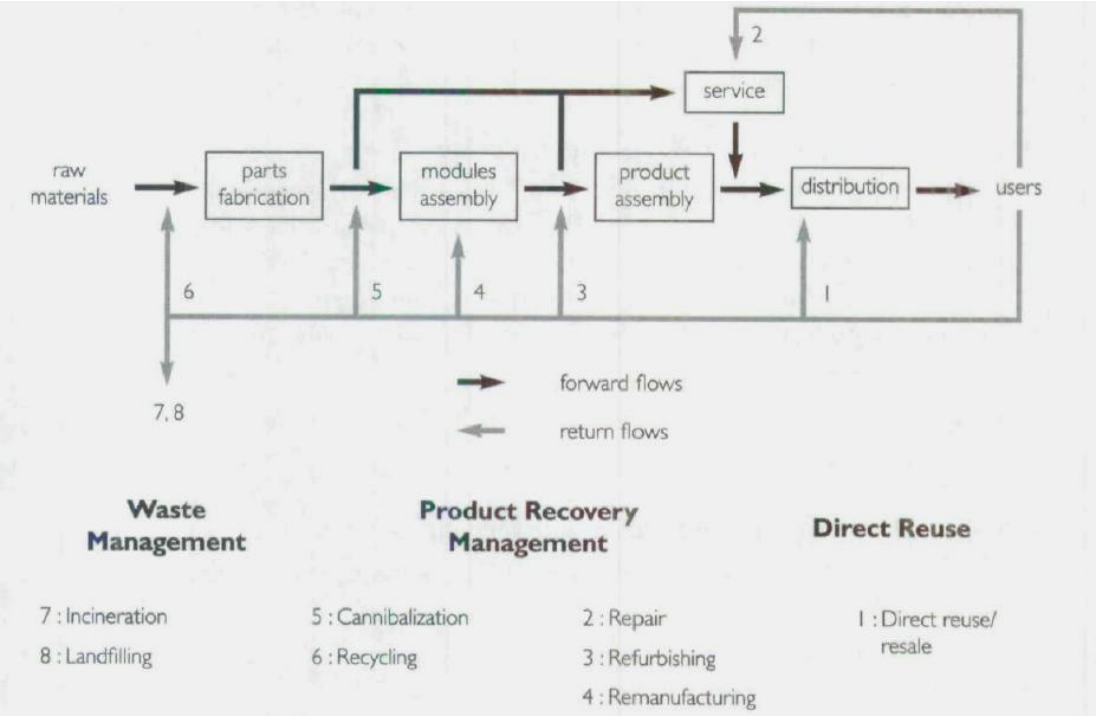

Figura 1.1 Supply chain integrata (tratto da Thierry *et al.*,1995)

[4] Possibili traduzioni del termine *refurbishing* possono essere: rimesso a nuovo, rinnovo, ristrutturazione, revisione, ricondizionamento. Qui si è scelto di usare il termine "revisione".

1.3.La gestione del recupero dei prodotti

La gestione del recupero dei prodotti (*Product Recovery Management*, PRM) condotta da un'industria manifatturiera riunisce tutte le attività di gestione dei prodotti, componenti e materiali utilizzati e scartati, che cadono sotto la responsabilità dell'azienda (Thierry *et al.*, 1995).
L'obiettivo del PRM è di avere, da tali prodotti, il maggior ritorno economico nel rispetto dell'ambiente, recuperando il valore che ancora possiedono e riducendo al massimo la quantità di rifiuti.
Con riferimento allo schema presentato in figura 1.1, si vuole descrivere le cinque opzioni di recupero individuate da Thierry *et al.*: riparazione, revisione, rilavorazione, cannibalizzazione e riciclaggio. L'ordine con cui sono elencate segue il grado (crescente) di disassemblaggio richiesto.

1.3.1.Riparazione (*repair*)

La riparazione ha lo scopo di ripristinare il funzionamento dei prodotti usati. Tale attività comporta la riparazione o la sostituzione delle parti danneggiate: gli altri componenti non sono sostanzialmente coinvolti.
La riparazione, dunque, richiede solo un parziale disassemblaggio e riassemblaggio del prodotto.
Inoltre, non essendo previsto un "aggiornamento tecnologico" delle parti, la qualità di un oggetto riparato è generalmente inferiore a quella posseduta dai prodotti nuovi. Le operazioni di riparazione possono essere effettuate presso il cliente o nei centri di riparazione controllati dal produttore.

1.3.2.Revisione (*refurbishing*)

La revisione permette ai prodotti usati di raggiungere il target di qualità specificato/richiesto. Tale attività comporta il disassemblaggio dell'oggetto in moduli, che vengono ispezionati e testati al fine di individuare, quindi riparare o sostituire, quelli critici. Alla fine, i moduli che superano il test vengono riassemblati in prodotti definiti "revisionati" o "ricondizionati".
Gli standard di qualità prefissati sono generalmente meno rigorosi rispetto quelli dei prodotti nuovi. Tuttavia, la revisione può essere abbinata all'ammodernamento della tecnologia se è previsto che le parti obsolete siano rimpiazzate con componenti "aggiornati".

Gli aerei, ad esempio, vengono revisionati con una certa frequenza. Tale operazione ne migliora significativamente la qualità e ne prolunga la rimanente vita utile.
Molte aziende elettroniche, informatiche e di telefonia offrono prodotti ricondizionati (ad esempio *Apple Inc.*[5]), di solito a prezzi convenienti.

1.3.3.Rilavorazione (*remanufacturing*)

La rilavorazione dei prodotti consente di ottenere standard di qualità rigorosi come quelli raggiunti dai prodotti nuovi. Tale attività comporta il completo disassemblaggio del prodotto prima in moduli e poi in parti, che quindi vengono ampiamente controllate, se necessario riparate oppure, nel caso siano usurate o obsolete, sostituite con componenti nuovi. Le parti funzionanti vengono prima pre-assemblate in moduli, che a loro volta sono assemblati in prodotti definiti "rilavorati" o "rifabbricati".
Anche la rilavorazione, come la revisione, può essere abbinata all'aggiornamento tecnologico.

1.3.4.Cannibalizzazione (*cannibalization*)

Nelle opzioni di recupero viste finora, la maggior parte dei componenti che costituiscono il prodotto usato, viene riutilizzata. Nella cannibalizzazione, solo una piccola parte viene recuperata. Tale attività, infatti, ha l'obiettivo di recuperare un set limitato di parti dai prodotti usati per adoperarlo nelle operazioni di riparazione, revisione e rilavorazione di altri beni.
La cannibalizzazione di prodotti usati comporta il disassemblaggio selettivo e la successiva ispezione delle parti potenzialmente riutilizzabili. I moduli e i componenti rimanenti a seguito della cannibalizzazione, dovranno seguire un altro percorso che sarà, presumibilmente, il riciclaggio o lo smaltimento.
Gli standard di qualità che devono raggiungere le parti cannibalizzate dipendono dal processo in cui saranno riutilizzate. Come intuibile, le parti destinate all'attività di rilavorazione devono soddisfare standard qualitativi più severi di quelle destinate alla revisione o alla riparazione.

[5] Nella pagina http://store.apple.com/it/browse/home/specialdeals/refurbfaq_popup è spiegato cosa si intende con "prodotti ricondizionati certificati Apple".

1.3.5.Riciclaggio (*recycling*)

L'obiettivo delle opzioni di recupero descritte finora, è quello di conservare l'identità e la funzionalità dei prodotti usati o dei loro componenti, per quanto ragionevolmente possibile. Con il riciclaggio, invece, si perdono l'identità e la funzionalità originali dei prodotti e dei componenti. Lo scopo di tale attività di recupero, infatti, è il riutilizzo dei materiali dei beni usati.

Il riciclaggio prevede il disassemblaggio del prodotto usato in parti, quindi la loro separazione in categorie distinte di materiali.

A seconda della qualità dei materiali recuperati, essi possono essere riutilizzati nella produzione di parti originali o in processi completamente diversi, oppure, in alternativa, venduti ad altre organizzazioni.

1.3.6.Confronto tra le diverse opzioni di recupero prodotto

Nella tabella 1.1 sono riassunte le diverse caratteristiche delle opzioni di recupero prodotto presentate precedentemente, evidenziando le principali differenze.

La figura 1.2 fornisce invece una vista completa delle attività contemplate nel PRM: la riparazione interviene già nella colonna 1, a livello di prodotto; la revisione avviene nella colonna 2, a livello di moduli; la rilavorazione inizia nella colonna 2 ma poi entra nella colonna 3 quando i moduli sono disassemblati in componenti e parti; la cannibalizzazione entra in gioco nella colonna 3; il riciclaggio nella colonna 4, a livello di materiali.

	Level of Disassembly	Quality Requirements	Resulting Product
Repair	To product level	Restore product to working order	Some parts fixed or replaced by spares
Refurbishing	To module level	Inspect all critical modules and upgrade to specified quality level	Some modules repaired/ replaced; potential upgrade
Remanufacturing	To part level	Inspect all modules and parts and upgrade to *as new* quality	Used and new modules/parts combined into *new* product; potential upgrade
Cannibalization	Selective retrieval of parts.	Depends on process in which parts are reused	Some parts reused; remaining product recycled/disposed
Recycling	To material level	High for production of original parts; less for other parts	Materials reused to produce new parts

Tabella 1.1 Comparazione tra le opzioni di recupero prodotto
(tratto da Thierry *et al.*,1995)

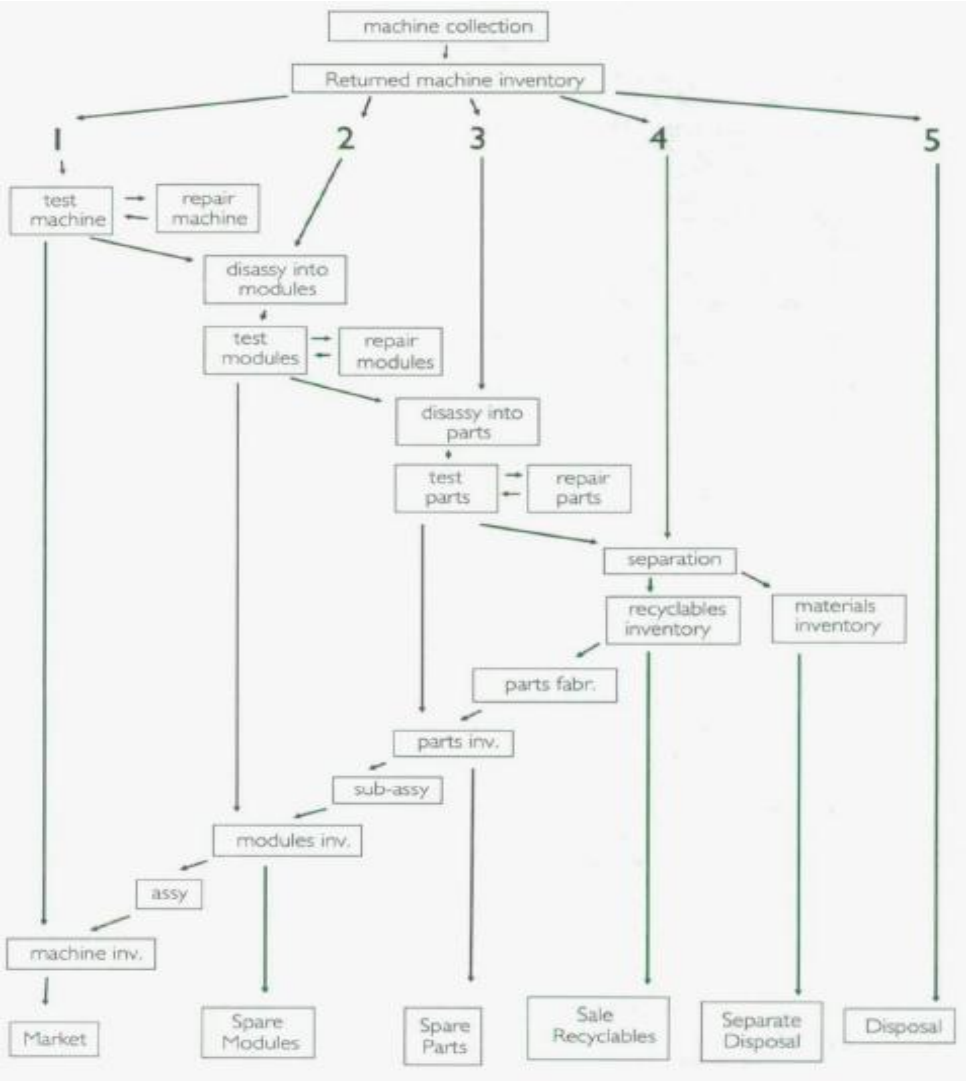

Figura 1.2 Visione integrata delle attività di PRM
(tratto da Thierry *et al.*,1995)

In figura 1.2 si vede inoltre come le opzioni di recupero interagiscono tra loro: per esempio, i moduli scartati dall'attività di rilavorazione possono tuttavia essere appropriati per la cannibalizzazione; oppure le parti eliminate dalla cannibalizzazione possono invece prestarsi al riciclaggio.

Infine, sempre in figura 1.2, sono indicate le possibili destinazioni di materiali, componenti (moduli e parti) e prodotti trattati. Ad esempio, parti e moduli recuperati possono essere usati per costruire altri moduli e prodotti, oppure impiegati come componenti di ricambio.

1.4.Modelli concettuali di CLSC

Negli ultimi anni il *Product Recovery Management* ha ricevuto crescente attenzione dovuta, come già spiegato, non solo ad una maggior sensibilità riguardo gli aspetti ambientali, ma anche ad incentivi economici e a fattori di costo (Francas e Minner, 2009).

Ovviamente, la letteratura scientifica è andata di pari passo, sviluppando molte ricerche collegate al tema del recupero di prodotti. Spesso gli articoli prendono come riferimento lo schema presentato in figura 1.1, rielaborandolo e riadattandolo.

In figura 1.3 è riportato il modello concettuale di CLSC adottato da Van Nunen e Zuidwijk (2004). Nello schema, pur non essendo previsto il riutilizzo diretto di un prodotto, vengono contemplate pressoché le stesse alternative di recupero individuate da Thierry *et al*., con l'aggiunta, all'interno del PRM, dell'opzione "pulizia e reimballaggio" (*clean and repackage*). Tale attività è destinata a prodotti in perfetto stato di funzionamento, che richiedono solamente una veloce pulizia superficiale per tornare come nuovi dal punto di vista estetico; prima di rivenderli, spesso è necessario rimetterli in nuovi imballaggi.

Effettivamente il riuso diretto è un'opzione improbabile nel caso di prodotti che sono stati utilizzati dai consumatori, ma diventa possibile se si tratta ad esempio di contenitori che girano all'interno di una supply chain.

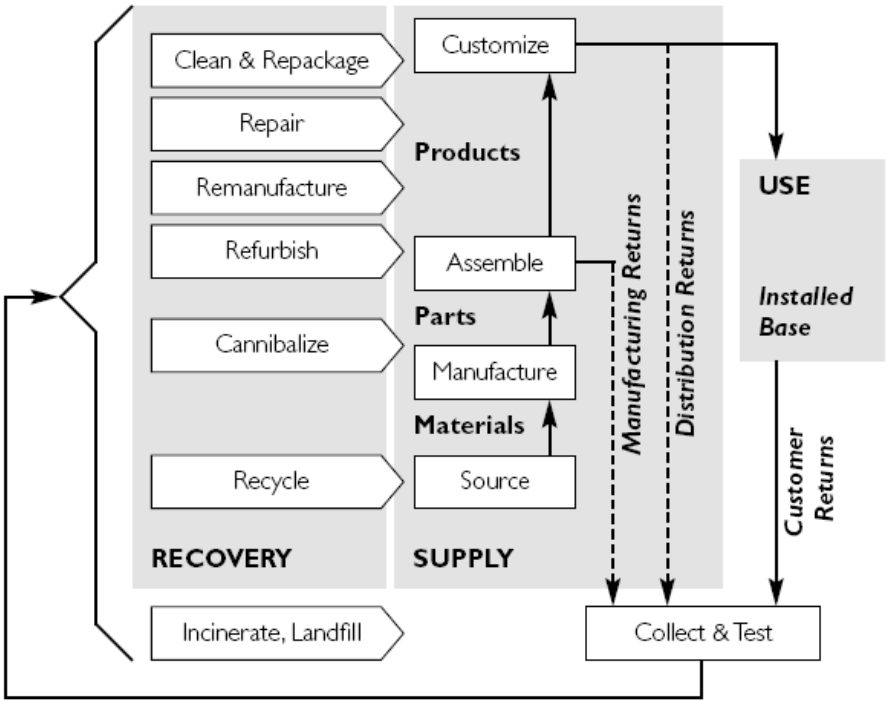

Figura 1.3 Modello concettuale di catena di fornitura a ciclo chiuso (tratto da Van Nunen e Zuidwijk, 2004)

In figura 1.3 sono evidenziati anche i tipi di prodotto di ritorno. Come si può vedere, il ritorno di materiale avviene praticamente in tutte le fasi del ciclo di vita del prodotto e, di conseguenza, i flussi inversi partono da diversi punti della supply chain.

Si hanno ritorni dovuti alle lavorazioni (*manufacturing returns*) a livello di fabbricazione e di assemblaggio dei prodotti: si tratta principalmente di surplus di materie prime, di pezzi che non superano qualche test di qualità e di scarti di produzione.

Poi si individuano i flussi di ritorno avviati dal canale di distribuzione (*distribution returns*) che comprendono i richiami di prodotti, i ritorni per ragioni commerciali e i beni obsoleti presenti nei magazzini di scorta.

I ritorni provenienti dai clienti (*customer returns*) riguardano le restituzioni di prodotti in garanzia, oppure che necessitano di assistenza/riparazione o che sono arrivati alla fine del loro uso da parte del consumatore.

Infine, si possono avere anche ritorni funzionali (*functional returns*), cioè prodotti o imballaggi che "rimbalzano" avanti e indietro nella catena di fornitura e che, in genere, non richiedono alcuna rilavorazione.

Anche Dekker *et al*. (2004), nel loro studio sui modelli quantitativi per le CLSC, ripropongono uno schema di supply chain in cui i flussi logistici di ritorno sono classificati secondo una prospettiva basata sui processi: anche qui si ritrovano le medesime opzioni di recupero individuate da Thierry *et al*., con il termine "cannibalizzazione" sostituito da "recupero di parti" (*parts retrieval*).

La letteratura scientifica recente si focalizza spesso sulle tematiche inerenti alla rilavorazione (*remanufacturing*) di componenti e al riciclaggio di materiale.

Di solito l'attività di rilavorazione è preferibile rispetto al riciclaggio, in quanto minimizza gli impatti ambientali e la perdita di valore, oltre a creare nuove opportunità di mercato. Tuttavia, la rilavorazione non è un'opzione fattibile per molte supply chain (Giuntini e Gaudette, 2003). In alcuni contesti, specialmente quelli che trattano beni di consumo, il costo della rilavorazione può superare il prezzo di vendita del prodotto nuovo. Soprattutto se il network della supply chain è caratterizzato da attori sparsi e mercati lontani, con costi elevati di movimentazione. Inoltre i mercati a cui sono destinati certi prodotti rilavorati, possono essere limitati: accade ad esempio nell'industria dell'elettronica, dove i cicli di vita dei prodotti sono brevi.

Sempre più spesso il canale della logistica inversa viene utilizzato per consentire il ritorno di materiali riciclabili (plastica, batterie, metalli, etc.), in alternativa allo smaltimento in discarica.

Pagell *et al.* (2007) evidenziano le implicazioni sulla supply chain per le aziende che si impegnano nella gestione del prodotto alla fine della sua vita utile (*end-of-life product*), nel caso in cui l'unica alternativa possibile sia il riciclaggio. Anche nel loro articolo si

parla di supply chain a ciclo chiuso (figura 1.4) per indicare la catena di fornitura di una compagnia che recupera il materiale usato dal consumatore.

La parte sotto della figura 1.4 illustra il flusso logistico inverso, che inizia con il recupero dei prodotti usati. Quando l'assistenza e la revisione (*service and refurbishment*) oppure la rilavorazione (*remanufacturing*) non sono strade percorribili, allora i prodotti recuperati entrano nel canale del riciclaggio. A questo punto si presentano due alternative:

- il riciclaggio dopo che il prodotto è stato disassemblato. I componenti recuperati vengono quindi indirizzati in diversi livelli della catena di fornitura diretta (parte sopra della figura 1.4) dove verranno riutilizzati, creando così un anello chiuso;
- il riciclaggio senza che il prodotto sia stato precedentemente disassemblato. I prodotti raccolti subiscono un approccio del tipo "*grind and sort*", cioè vengono prima compattati o triturati e poi suddivisi per tipo di materiale. La destinazione finale può essere un mercato secondario o, nello scenario peggiore, l'incenerimento o lo smaltimento in discarica.

Spesso la seconda strategia è perseguita quando l'azienda vuole minimizzare i costi o non vuole rischiare di squilibrare la catena di fornitura; tuttavia a bassi rischi corrisponderanno bassi ritorni nel futuro. D'altro canto, implementare un sistema di riciclaggio che prevede il disassemblaggio, inizialmente costerà di più perché necessita di investimenti in attrezzature dedicate ad ogni famiglia di prodotti e, probabilmente, anche di manodopera qualificata (Pagell *et al.,* 2007).

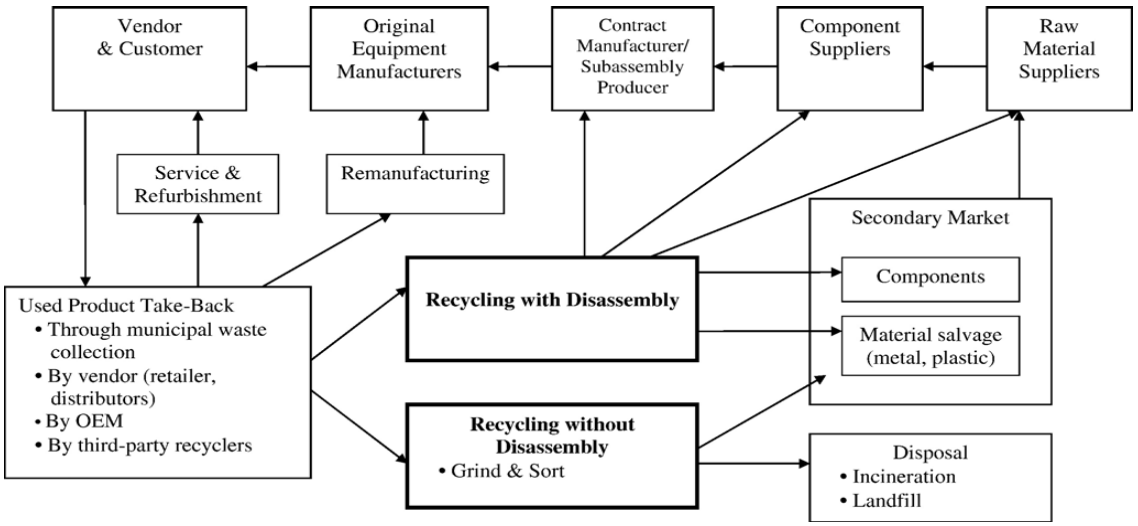

Figura 1.4 CLSC con alternative di riciclaggio
(tratto da Pagell *et al.*, 2007)

1.5.Attività di recupero di prodotto e tipi di ritorno

Le supply chain a ciclo chiuso possono essere classificate dal punto di vista dei tipi di ritorno che si verificano al loro interno o focalizzandosi sulle attività di recupero precedentemente descritte.

Sarebbe utile stabilire un collegamento tra le due prospettive: una soluzione in merito è proposta da Guide e Van Wassenhove (2009). Innanzitutto gli autori raggruppano in tre categorie i possibili ritorni di prodotto:

- ritorni dovuti a ragioni commerciali (*commercial returns*): sono i prodotti restituiti dai consumatori ai rivenditori a seguito di politiche commerciali del tipo "soddisfatti o rimborsati";
- ritorni a fine uso (*end of use returns*): sono prodotti funzionanti che vengono rimpiazzati da articoli tecnologicamente più recenti;
- ritorni a fine vita utile (*end of life returns*): sono prodotti tecnologicamente obsoleti che non sono più di nessuna utilità per il consumatore.

È dunque possibile associare i tipi di ritorni appena elencati ad una specifica attività di recupero: per ogni tipo di ritorno esiste, infatti, un'opzione più conveniente di recupero (Guide e Wassenhove, 2009).

I ritorni dovuti a ragioni commerciali coinvolgono prodotti a malapena usati e vanno reintrodotti nel mercato il più rapidamente possibile. La maggior parte di essi necessita di veloci operazioni di ripristino (pulizia e ritocchi superficiali, seguiti dal reimballaggio).

I ritorni a fine d'uso sono prodotti che, generalmente, sono stati usati intensamente e quindi necessitano di attività di rilavorazioni più profonde: a causa dell'elevata variabilità nel loro livello di utilizzo, essi possono richiedere disposizioni molto diverse tra loro.

I prodotti arrivati a fine vita utile sono obsoleti e spesso usurati, quindi destinati al riciclaggio (oppure, nel caso peggiore, allo smaltimento).

Ricapitolando, le coppie dominanti che si riscontrano nella realtà industriale sono:

- ritorni per ragioni commerciali → pulizia e reimballaggio;
- ritorni a fine d'uso → riparazione, revisione, rilavorazione;
- ritorni a fine vita utile → riciclaggio (o smaltimento).

Sviluppando ulteriormente i ragionamenti di Guide e Van Wassenhove si possono trovare altre coppie, come:

- ritorni di contenitori → riuso diretto
- ritorni di prodotti in garanzia → riparazione
- ritorni di prodotti noleggiati → veloce pulizia prima del riuso (e riparazione programmata)

Anche Lamsali e Liu (2008) hanno affrontato il problema che concerne la selezione dell'opzione di recupero adatta per i prodotti di ritorno. A tale fine hanno sviluppato un modello di programmazione lineare per classificare i prodotti di ritorno in diverse classi di qualità per opzione di recupero. Il modello, che ha come funzione obiettivo la massimizzazione del profitto, considera la qualità e la quantità dei prodotti di ritorno, la domanda e il prezzo dei prodotti recuperati e i costi per tipo di recupero.

1.6.Modelli quantitativi per lo studio di network logistici inversi

Nell'ultimo decennio diversi autori si sono impegnati nello sviluppare modelli quantitativi al fine di studiare i network logistici inversi: Wang e Bai (2010) li hanno catalogati sulla base della tipologia di programmazione adottata (tabella 1.2) e della categoria di network considerata (tabella 1.3).
Nel seguito si richiamano i lavori di alcuni degli autori sopra menzionati: per un'analisi più approfondita si rimanda all'articolo: "*Reverse Logistics Network: A Review*" (Wang e Bai, 2010).

- Fleischmann *et al*. (2001) hanno sviluppato un modello per analizzare l'impatto che il recupero dei prodotti ha sulla rete logistica.
- Kusumastuti *et al*. (2004) hanno presentato un modello multi-obiettivo e multi-periodo per la progettazione del network logistico inverso, che determina il numero di strutture da usare tra quelle già esistenti nella catena di fornitura diretta e il numero di strutture dedicate da installare per gestire il flusso di ritorno.
- Kara *et al*. (2007) hanno simulato un modello di logistica inversa per la raccolta dei prodotti a fine vita.
- Salema *et al*. (2007) hanno proposto un generico modello di rete logistica inversa che considera contemporaneamente i limiti di capacità, l'incertezza sulla domanda e sul ritorno di prodotti.
- Kannan *et al*. (2010) hanno sviluppato un modello di programmazione lineare per determinare il livello di materie prime, di produzione, di smaltimento e di riciclo nelle strutture di una supply chian a ciclo chiuso, con l'obiettivo di minimizzare i costi totali.

Quantity model	Paper
Mixed integer linear programming (MILP)	Jayaraman et al. (1999), Krikke et al. (1999), Fleischmann et al (2001), Shih et al. (2001), Jayaraman et al. (2003), Schultmann et al. (2003), Kusumastuti et al. (2004), Realff et al. (2004), Listes et al. (2005), Listes et al. (2007), Lu et al. (2007), Salema et al. (2007), El-Ssayed et al . (2008), Zhou et al. (2008), Du et al. (2008), Min et al. (2008), Xanthopoulos et al. (2009), Kannan et al. (2010), Pishvaee et al. (2010), Salema et al. (2010)
Mixed integer nonlinear programming (MINLP)	Min et al. (2005), Min et al. (2006), Min et al. (2006), Ko et al.(2007), Lieckens et al. (2007), Aras et al.(2008)
Mixed integer goal programming (MIGP)	Pati et al. (2008)
Linear multi-objective programming (LMOP)	Sheu et al. (2005), Sheu et al. (2008)

Tabella 1.2 Modelli quantitativi suddivisi per tipologia di programmazione (tratto da Wang e Bai, 2010)

Category	Paper
Closed-loop	Jayaraman et al. (1999), Fleischmann et al. (2001), Schultmann et al. (2003), Beamon et al. (2004), Min et al. (2005), Sheu et al. (2005), Schultmann et al. (2006), Min et al. (2006), Listes et al. (2007), Ko et al. (2007), Lu et al. (2007), Salema et al. (2007), Lee et al. (2008), EI-Sayed et al. (2008), Sheu et al. (2008), Fuente et al. (2008), Kusumastuti et al. (2008), Reynaldo et al. (2009), Pishvaee et al. (2010), Salema et al. (2010), Kannan et al. (2010)
Generic model	Fleishmann et al. (2001), Listes et al. (2007), Salema et al. (2007), Zhou et al. (2008), Salema et al. (2010)
Stochastic model	Hinojosa et al. (2001), Inderfuth et al. (2001), Listes et al. (2005), Listes et al. (2007), Salema et al. (2007), EI-Sayed et al. (2008), Min et al. (2008), Lee et al. (2009), Qin et al. (2010)
3PLs	Ko et al. (2007), Du et al. (2008), Min et al. (2008)

Tabella 1.3 Modelli quantitativi suddivisi per categoria di network (tratto da Wang e Bai, 2010)

La maggior parte dei modelli quantitativi presenti in letteratura si focalizza sulla progettazione (o sulla riprogettazione) del network, considerando simultaneamente la catena di fornitura diretta e quella inversa.

Nonostante gli autori ricorrano spesso a semplificazioni e ad assunzioni di partenza, si tratta comunque di modelli complessi, che richiedono la conoscenza di numerosi input e che hanno come driver di analisi il fattore costo. Alcuni di questi tengono conto anche dell'incertezza associata alla quantità e alla qualità dei prodotti di ritorno.

Tuttavia non sono presenti in letteratura modelli matematici per lo studio dei flussi logistici che abbiano l'obiettivo di ricavare un indice capace di sintetizzare l'entità dei ricircoli presenti in una CLSC.

1.7.Esternalizzazione e delocalizzazione

Oggigiorno le tecnologie dell'informazione e della comunicazione (*Information and Communication Technology*, ICT) applicate in un contesto economico sempre più globale, creano nuove opportunità per le compagnie per quanto riguarda l'esternalizzazione (*outsourcing*) delle loro attività tradizionali (Guo *et al*., 2010) o la delocalizzazione della produzione.

L'esternalizzazione può prevedere un contratto di subfornitura, nel quale l'impresa committente si avvale di una impresa fornitrice per la produzione di prodotti finiti o semilavorati. Spesso tale pratica prevede che del materiale sia consegnato in "conto-lavoro" all'azienda fornitrice, la quale esegue un processo di lavorazione e riconsegna il prodotto semilavorato o finito all'azienda committente.

Ovviamente tali scelte comportano l'insorgere di flussi logistici di ricircolo all'interno della supply chain. La letteratura scientifica, pur riconoscendo la pratica del conto-lavoro all'interno dei suoi studi, non la contempla dal punto di vista della logistica inversa e di conseguenza, non approfondisce l'aspetto riguardante i flussi di ricircolo annessi.

Con l'obiettivo di identificare tutte le categorie di ricircolo, in questa sede viene adoperato il generico termine "conto-lavoro" al fine di classificare i ricircoli appena presentati.

1.8.Riepilogo

Dekker *et al*. (2004) descrivono la logistica inversa da quattro punti di vista che rispondono a domande diverse:

1) *Perché* ci sono prodotti e componenti che ritornano indietro lungo la catena di fornitura? Le motivazioni possono essere di stampo economico, legislativo, ambientale ed etico.

2) *Come* vengono processati i ritorni? La risposta fa riferimento alle attività effettuate nei processi della logistica inversa che comprendono anche le opzioni di recupero.

3) *Cosa* ritorna? Implica classificare i prodotti/componenti secondo la tipologia (beni di consumo, scarti industriali, prodotti chimici, imballaggi, ...);

4) *Chi* esegue le attività della logistica inversa? Gli attori possono essere quelli del canale diretto (produttori, grossisti, *retailer*) oppure attori specializzati nella logistica inversa.

Al fine di individuare le tipologie di ricircolo presenti nelle supply chain, nel corso della trattazione del capitolo è stato assunto il secondo punto di vista, che coincide con la prospettiva basata sui processi. All'interno delle attività in cui si articola il processo della logistica inversa (raccolta, ispezione/selezione/ordinamento, trattamento e redistribuzione), la discriminante che permette di classificare i ricircoli è il tipo di trattamento effettuato. Si individuano tre gruppi di trattamento, di seguito richiamati.

Il primo coinvolge i prodotti di ritorno in condizioni praticamente nuove e quindi consente un recupero veloce attraverso:

- riuso/rivendita/redistribuzione diretti.

Il secondo gruppo concerne le opzioni di recupero che richiedono delle attività specifiche e che consentono di recuperare il prodotto a livelli diversi:

- pulizia e reimballaggio (*clean and repackage*) → a livello di prodotto (senza passare per il suo disassemblaggio);
- riparazione (*repair*) → a livello di prodotto (è previsto un suo parziale disassemblaggio);
- revisione (*refurbishing*) → a livello di moduli;
- rilavorazione (*remanufacturing*) → a livello di componenti;
- recupero parti (*parts retrieval*) → a livello (selettivo) di parti;
- riciclaggio (*recycling*) → a livello di materiali;

Infine, il terzo gruppo, non consente un recupero, ma riguarda la gestione dei rifiuti attraverso:

- incenerimento o smaltimento in discarica.

All'interno della supply chain si possono inoltre avere flussi logistici di ricircolo dovuti a:

- conto-lavoro.

CAPITOLO 2

Indici per l'analisi dei network

2.1.Network analysis

L'obiettivo principale di questo lavoro di tesi è analizzare i flussi logistici di ricircolo esistenti all'interno di un supply network e ricavare dei parametri che forniscano un quadro completo riguardo il fenomeno.

Tale ricerca si inserisce in un contesto più ampio che concerne lo studio (e la misurazione) della complessità delle catene di fornitura da un punto di vista olistico[6]. La aziende infatti non possono più permettersi di rimanere isolate e di focalizzarsi solo sui processi interni all'impresa: la loro competitività dipende dall'efficienza della supply chain nel suo complesso. Esse devono quindi adottare una visione globale al fine di ottimizzare l'intera catena di fornitura e ricavare valore per tutti gli attori coinvolti.

Battini *et al*. (2006) propongono una nuova metodologia per l'ottimizzazione e il monitoraggio delle supply network, basata sul modello entropico[7]. La loro ricerca si fonda sull'analisi dei network (*network analysis*), pratica diffusa soprattutto nell'ambito dell'ecologia per lo studio degli ecosistemi naturali.

Un ecosistema è costituito da catene e reti alimentari: si tratta di raggruppamenti di piante e specie animali organizzate in complicate strutture a ragnatela (come quelle dei siti web), nelle quali l'energia e la materia vengono trasferite e trasformate (Battini *et al*., 2006). Tali strutture sono descritte da network e rappresentate mediante grafi orientati che descrivono le interazioni alimentari tra le diverse specie presenti. Nei grafi si individuano i nodi, cioè i vari compartimenti (animali o altre specie), collegati da archi che riproducono gli scambi (flussi di energia e di materia) che avvengono all'interno del sistema e con l'ambiente esterno.

L'analisi dei network ecologici (*ecological network analysis*, ENA) consiste in una serie di strumenti e procedure a disposizione dei ricercatori, tesi a testare il grado di

[6] Il punto di vista olistico studia i sistemi complessi nella loro globalità, in contrasto con l'approccio analitico che esamina i sistemi complessi dividendoli nelle loro componenti che poi studia separatamente.

[7] I modelli entropici sono modelli matematici che derivano dall'entropia dell'informazione, teoria di cui Shannon e Weaver (1948) sono i precursori.

organizzazione di un ecosistema, analizzarne i percorsi interni, valutare il numero di livelli trofici[8], stimare gli effetti indiretti e altro ancora (Allesina, 2004).

Nello specifico, gli strumenti usati nell'ENA per descrivere la struttura di un network, si dividono in:

1) rappresentazione del network e stato di equilibrio (*steady state*)
2) analisi input/output
3) analisi dei livelli trofici
4) analisi dei cicli
5) calcolo degli indici di prestazione

Con questi strumenti gli ecologisti sono riusciti a ricavare degli indici in grado di quantificare gli attributi di un ecosistema nella sua globalità. Tali misure, opportunamente adattate, si possono applicare anche per studiare le performance delle supply chain industriali.

Esistono infatti delle analogie tra i network degli ecosistemi e i network industriali (Allesina *et al.*, 2010).

Una rete di fornitura industriale è composta da molteplici attori che coprono una gamma di attività (dall'approvvigionamento di materia prima, all'assistenza dei clienti), determinando il trasferimento e la trasformazione di energia, informazioni e beni/servizi lungo tutta la catena. Questi processi creano delle connessioni funzionali che collegano le attività l'una all'altra in una struttura a ragnatela, del tutto simile a quella dei network ecologici.

Di conseguenza, anche le supply chain si possono rappresentare mediante grafi orientati e studiare applicando le teorie e gli strumenti derivanti dalla network analysis.

Il seguito del capitolo è strutturato in questo modo: nel paragrafo 2.2 viene descritta modalità di rappresentazione di una supply chain industriale mediante grafo orientato, approccio che permette di mappare gli scambi di beni tra i diversi attori del network.

Nel paragrafo 2.3 vengono riportati otto indici ripresi direttamente dal lavoro di Allesina *et al.* (2010): si tratta di indicatori che derivano dall'analisi dei network ecologici, applicati alle reti di fornitura industriali, per misurarne la complessità e il livello di organizzazione.

Nel paragrafo 2.4 vengono infine analizzati i ricircoli, vero focus di questo lavoro, nel seguente ordine: all'inizio si riporta qualche accenno sulle origini dell'analisi e sui primi studi condotti in ambito ecologico; successivamente si presenta la metodologia, opportunamente adattata ad un caso industriale, che permette di calcolare un

[8] Il livello trofico è il posto occupato da un individuo all'interno della catena alimentare.

parametro in grado di stimare l'entità dei ricircoli; infine si propone un'ulteriore analisi per fornire un quadro più esaustivo sul fenomeno dei ricircoli logistici.
Nel seguente paragrafo viene inoltre presentato un caso studio che farà da filo conduttore lungo tutto il capitolo.

2.2.Rappresentazione del network di una supply chain

Il network di una supply chain si può rappresentare mediante un grafo orientato contenente:

- nodi, che corrispondono agli attori della supply chain (fornitori, OEM, *retailer*, etc.) oggetto di analisi;
- archi, per indicare i flussi in entrata (cioè quelli che provengono dall'ambiente esterno al network e arrivano ai nodi), i flussi in uscita (che partono dai nodi ed escono dal sistema), i flussi di scambio (collegamenti tra gli attori della supply chain) e le dissipazioni (dispersioni dei nodi);
- pesi associati agli archi, per indicare la dimensione dei flussi.

Ovviamente, bisogna prima individuare la tipologia di flussi che si vuole rappresentare (ad esempio prodotti, materiali vari, pezzi, unità di carico, camion, denaro, etc.) e una unità di misura idonea ad esprimerne il peso (tonnellate di materiale annue, numero di pezzi al mese, etc.).
In figura 2.1 è schematizzato il supply network di un semplice caso industriale che coinvolge sette differenti attori, di cui tre fornitori di componenti o sub-assemblati (S1, S2, S3), un produttore (M) presso cui viene assemblato il prodotto finale, due *retailer* (C1, C2) e un centro di raccolta (R) per il riciclaggio di materiale. I valori riportati sugli archi indicano la quantità di materiale scambiato, misurato in tonnellate/anno.
Si possono, quindi, identificare:

- il vettore I dei flussi in entrata (*import vector*): si tratta, ad esempio, di materie prime provenienti dall'ambiente esterno al network, che alimentano i fornitori durante tutto il periodo di riferimento;
- il vettore E dei flussi in uscita (*export vector*): è il materiale che viene riusato all'esterno del sistema;
- il vettore D delle dissipazioni (*dissipations vector*): può indicare, ad esempio, l'entità degli scarti di una lavorazione effettuata presso un fornitore;
- la matrice T dei trasferimenti (*transfers matrix*): contiene gli scambi di beni tra gli attori all'interno della supply chain considerata.

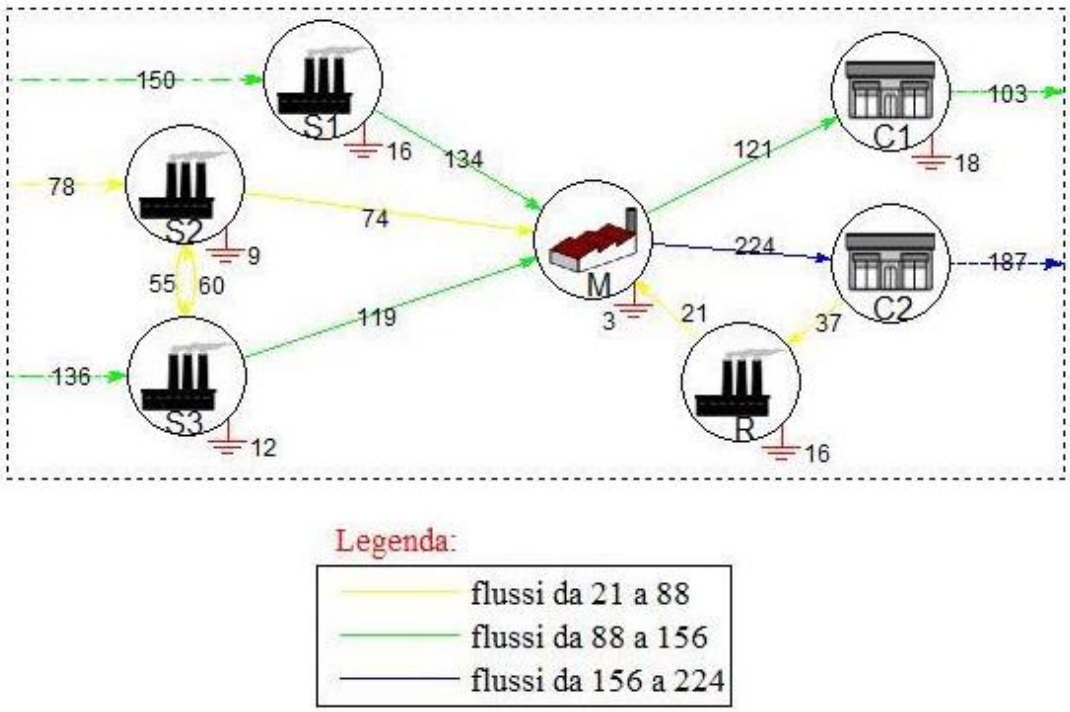

Figura 2.1 Rappresentazione di una supply chain industriale

A questo punto, si possono riunire in un'unica matrice tutte le informazioni sui flussi che si verificano in un network, creando la matrice estesa dei trasferimenti (*extended transfers matrix*, T^*). Essa riporta gli scambi che avvengono all'interno del sistema e tra quest'ultimo e l'ambiente esterno (tabella 2.1).

In tabella 2.2 è riportata la matrice estesa dei trasferimenti T^*, associata al grafo orientato in figura 2.1.

	0	1	2	…	N	N+1	N+2
0	0	Input [I]				0	0
1	0	Trasferimenti tra i compartimenti [T]				Export [E]	Dissipation [D]
2	0						
…	0						
N	0						
N+1	0	0	0	0	0	0	0
N+2	0	0	0	0	0	0	0

Tabella 2.1 Matrice estesa dei trasferimenti T^*

	Imp	S1	S2	S3	M	C1	C2	R	Exp	Diss
Imp		150	78	136						
S1					134					16
S2				55	74					9
S3			60		119					12
M						121	224			3
C1									103	18
C2								37	187	
R					21					16
Exp										
Diss										

Tabella 2.2 Matrice estesa dei trasferimenti T^* del network in figura 2.1

2.2.1.Stato di equilibrio (*steady state*)

L'analisi dei network esamina le proprietà di una rete industriale focalizzandosi sulle relazioni tra i diversi attori. Questa metodologia, ripresa dall'ecologia, richiede che ogni compartimento sia in stato di equilibrio (*steady state*).

In sostanza, per ogni nodo i deve valere il bilancio delle masse: significa che il flusso di materiale in entrata nel compartimento deve corrispondere al flusso in uscita dallo stesso, al netto delle perdite, in modo che non ci sia variazione della massa totale. Ciò è esprimibile mediante la seguente uguaglianza:

$$T_{.i} + I_i = T_{i.} + E_i + D_i \,, \qquad \forall i$$

dove $T_{.i}$ rappresenta la somma della i-esima colonna, mentre $T_{i.}$ la somma della i-esima riga. Il primo e il secondo membro della precedente equazione individuano, rispettivamente, il vettore dei flussi entranti (*inflow vector*, S) e il vettore dei flussi uscenti (*outflow vector*, $S^{'}$).

Se il bilancio delle masse è verificato per ogni nodo del network (dunque se $S = S^{'}$), allora l'intero sistema nel suo complesso si trova in stato di equilibrio (*steady state*).

Tuttavia, quando si procede con la raccolta dei dati per la costruzione di un network, è improbabile che i flussi risultino perfettamente bilanciati. Per quanto riguarda il calcolo degli indici entropici (derivanti dalla teoria dell'informazione (Shannon e Weaver, 1948)), il bilancio delle masse non è indispensabile, ma nello studio dei ricircoli la condizione di *steady state* risulta fondamentale.

In ambito ecologico sono già stati sviluppati diversi algoritmi per bilanciare i network (Allesina, 2004): si tratterebbe di vedere quale tra questi metodi si adatta maggiormente alle caratteristiche delle reti industriali. Questo lavoro di tesi, che ha

l'obiettivo principale di introdurre una nuova metodologia di analisi dei network e dimostrarne la fattibilità, non affronta il problema del bilanciamento dei dati grezzi.

2.3.Calcolo degli indici entropici

Allesina *et al*. (2010) propongono l'uso di otto indici, basati sul concetto di entropia, in grado di riassumere il livello di complessità del network di fornitura nella sua globalità. Per calcolarli è sufficiente disporre della matrice estesa dei trasferimenti T^*. Gli indici entropici, descritti più approfonditamente nei seguenti sottoparagrafi, sono:

- il *total system throughput* (TST);
- *l'average mutual information* (AMI);
- l'*ascendancy* (ASC);
- il *development capacity* (C);
- l'*overhead* (Φ), costituito da quattro componenti, tra i quali il più importante è il *redundancy* (R).

2.3.1.*Total System Throughput* (TST)

Il *total system throughput* fornisce un'indicazione sull'estensione dell'intero sistema o sull'ammontare di beni[9] che fluisce attraverso il network. Si calcola con la seguente formula (che esprime la somma tutti i coefficienti della matrice estesa T^*):

$$TST = \sum_{i=0}^{N+2}\sum_{j=0}^{N+2} T^*_{ij} = t_{\cdot\cdot} \qquad \text{con } t \in T^*$$

dove i puntini a pedice di $t_{\cdot\cdot}$ stanno ad indicare la somma lungo le righe (primo puntino) e lungo le colonne (secondo puntino).

Per il network in figura 2.1 la somma dei flussi è pari a: $TST = 1573$ ton/anno.

Dato che si basa sul numero di nodi (attori) e sull'entità dei flussi (beni trasferiti nel sistema), l'indice TST è una misura direttamente collegata allo sviluppo del network. Tuttavia, non fornisce informazioni sulla distribuzione dei beni all'interno del sistema: supply chain aventi stesso TST possono essere caratterizzate da configurazioni di flusso totalmente differenti.

---

[9] Per una questione di semplicità, nel seguito del capitolo si fa riferimento, in modo intercambiabile, a flussi di beni, prodotti o materiali, pur sapendo che l'analisi si può estendere ad altre tipologie di flussi (pezzi, UDC, camion, informazioni, denaro, etc.).

2.3.2. *Average Mutual Information* (AMI)

L'*average mutual information* è un parametro entropico che consente di stimare il livello di organizzazione (concetto opposto alla complessità) di un network. Per calcolare l'AMI si deve partire dal lavoro di Shannon e Weaver (1948) che per primi misurarono l'entropia associata ad un processo (evento) X come somma dei prodotti tra la probabilità $p(i)$ di ogni output i e il rispettivo logaritmo[10]:

$$H_X = -\sum_{i \in X} p(i) \cdot \log p(i)$$

Il network di una supply chain può essere visto come un insieme di transizioni (eventi), ognuna avente una certa probabilità, tra i diversi attori.

La probabilità che un prodotto (o una tonnellata di materiale, o un pezzo, o una unità di carico, o un camion, etc.) vada da un compartimento i ad un compartimento j è proporzionale al flusso che va da i a j:

$$p_{I,O}(i,j) = \frac{t_{ij}}{t_{..}} \qquad \text{con } t \in T^*$$

È dunque possibile esprimere l'entropia dell'intero sistema considerando, per ogni nodo, i rispettivi flussi in entrata (input) e in uscita (output).

Innanzitutto si calcola la *joint entropy*, definita come la somma delle probabilità che un prodotto lasci il nodo i ed entri nel nodo j:

$$H_{I,O} = -\sum_{i=0}^{N+2}\sum_{j=0}^{N+2} p_{I,O}(i,j) \log p_{I,O}(i,j)$$

$$H_{I,O} = -\sum_{i=0}^{N+2}\sum_{j=0}^{N+2} \frac{t_{ij}}{t_{..}} \log \frac{t_{ij}}{t_{..}}$$

Per quanto riguarda l'entropia associata ai flussi in uscita da ogni nodo (entropia in output), è pari a:

$$H_O = -\sum_{i=0}^{N+2} p_O(i) \log p_O(i)$$

$$H_O = -\sum_{i=0}^{N+2} \frac{t_{i.}}{t_{..}} \log \frac{t_{i.}}{t_{..}}$$

dove $t_{i.}$ corrisponde alla somma dei coefficienti lungo la riga i della matrice T^*.

---

[10] La scelta della base del logaritmo determina l'unità di misura da associare all'entropia. Nel contesto della teoria dell'informazione, la base usata è due: il risultato si può quindi esprimere in bit. Nel seguito della trattazione si sottintende sempre logaritmo in base due.

Infine, l'entropia associata ai flussi in entrata verso ogni nodo (entropia in input) è uguale a:

$$H_I = -\sum_{j=0}^{N+2} p_I(j) \log p_I(j)$$

$$H_I = -\sum_{i=0}^{N+2} \frac{t_{.j}}{t_{..}} \log \frac{t_{.j}}{t_{..}}$$

dove $t_{.j}$ corrisponde alla somma dei coefficienti lungo la colonna j della matrice T^*.

A questo punto è possibile trovare l'*average mutual information* (AMI), definito come la somma delle entropie in input e in output, diminuita della *joint entropy*:

$$AMI = H_O + H_I - H_{I,O}$$

Tornando al caso studio in figura 2.1, il contributo di ogni coefficiente della relativa matrice T^* (tabella 2.2) alla *joint entropy* è dato da $\left(-\frac{t_{ij}}{t_{..}} \log \frac{t_{ij}}{t_{..}} \right)$, con $t \in T^*$.

In tabella 2.3 sono riportati tutti i contributi che, sommati, danno una *joint entropy* pari a: $H_{I,O} = 3.825$ bit.

Per calcolare l'entropia in input H_I si devono prima sommare tutti i valori della matrice estesa T^* (tabella 2.2) lungo ogni colonna j, in modo da ricavare $t_{.j}$ per ogni nodo j. Il contributo del nodo j sarà quindi $\left(-\frac{t_{.j}}{t_{..}} \log \frac{t_{.j}}{t_{..}} \right)$ che, unito agli altri, dà l'entropia in input: $H_I = 2.952$ bit.

Allo stesso modo, ma questa volta sommando i coefficienti lungo ogni riga i in modo da ottenere $t_{i.}$, si trovano i contributi $\left(-\frac{t_{i.}}{t_{..}} \log \frac{t_{i.}}{t_{..}} \right)$ che, uniti, danno l'entropia in output: $H_O = 2.783$ bit.

	Imp	S1	S2	S3	M	C1	C2	R	Exp	Diss
Imp	0	0,323	0,215	0,305	0	0	0	0	0	0
S1	0	0	0	0	0,303	0	0	0	0	0,067
S2	0	0	0	0,169	0,208	0	0	0	0	0,043
S3	0	0	0,180	0	0,282	0	0	0	0	0,054
M	0	0	0	0	0	0,285	0,400	0	0	0,017
C1	0	0	0	0	0	0	0	0	0,258	0,074
C2	0	0	0	0	0	0	0	0,127	0,365	0
R	0	0	0	0	0,083	0	0	0	0	0,067
Exp	0	0	0	0	0	0	0	0	0	0
Diss	0	0	0	0	0	0	0	0	0	0

Tabella 2.3 Matrice dei contributi alla *joint entropy*

É dunque possibile calcolare l'indice l'AMI per il network in figura 2.1:

$$AMI = H_O + H_I - H_{I,O}$$

$$AMI = 2.783 + 2.952 - 3.825 = 1.910 \text{ bit}$$

L'AMI misura il grado di organizzazione di un network o, in altre parole, quanto sono vincolati i percorsi dei flussi al suo interno.

Le configurazioni più vincolate sono quelle che presentano un numero limitato di strade percorribili da un prodotto che si muove lungo la supply chain: questo si verifica quando gli attori sono altamente specializzati in determinate funzioni.

In figura 2.2 viene illustrato il significato dell'AMI mediante diagrammi di Venn: H_O e H_I sono rappresentati da cerchi che si sovrappongono, mentre l'AMI corrisponde proprio alla loro intersezione.

In una configurazione in cui ogni nodo è connesso con tutti gli altri e i flussi sono uguali, l'AMI è pari a zero: in questa circostanza i due cerchi risultano staccati. Tradotto, significa che conoscere la posizione di un prodotto all'interno del network non fornisce alcuna informazione sulla sua destinazione successiva.

Nel caso opposto, in una configurazione in cui i flussi sono completamente vincolati, i due cerchi risultano completamente sovrapposti. In questa situazione, sapere che un prodotto si trova presso un certo attore della rete di fornitura implica conoscere anche il nodo di destinazione.

Valori alti di AMI si hanno solo nelle strutture in cui il flusso di materiale è vincolato in determinati percorsi. Dunque, supply chain altamente organizzate, in cui la movimentazione delle merci avviene lungo poche strade efficienti (con bassi costi di gestione dell'intero sistema), sono caratterizzate da valori elevati di AMI. D'altro canto, network poco organizzati presentano flussi altamente ridondanti e perciò possiedono valori di AMI più bassi.

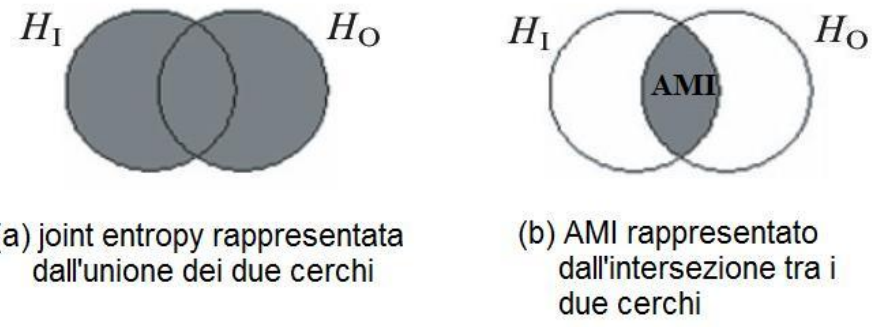

Figura 2.2 Diagrammi di Venn che esprimono la relazione tra le entropie H_O e H_I e l'indice AMI (tratto da Allesina *et al*., 2010)

2.3.3.*Ascendancy* (A)

Essendo l'AMI una misura adimensionale, Ulanowicz e Kay (1991) proposero di pesarla con la somma di tutti i flussi (TST). Dal prodotto di AMI per TST si ottiene l'indice chiamato *ascendancy*:

$$A = TST \times AMI$$

$$A = \sum_{i=0}^{N+2}\sum_{j=0}^{N+2} t_{ij} \log\left(\frac{t_{ij}t_{..}}{t_{i.}t_{.j}}\right)$$

Per il network in figura 2.1 si ha: $A = 3004{,}7$ bit.

L'*ascendancy* è in grado di valutare il livello di sviluppo di un sistema, considerando sia la sua dimensione (TST), sia il suo grado di organizzazione (AMI).

Calcolato per una supply chain, l'*ascendancy* ne misura la frazione di beni distribuiti in modo efficiente al suo interno.

2.3.4.*Development Capacity* (C)

Nel campo dell'ecologia, dove sono nati gli indici qui presentati, valori elevati di *ascendancy* rappresentano una rete alimentare matura, in cui i compartimenti (animali o altre specie) sono specializzati, gli scambi sono strutturati e i trasferimenti sono efficienti. Nel caso di un ecosistema che abbia raggiunto il suo massimo potenziale di sviluppo e di organizzazione, l'*ascendancy* è uguale alla sua massima capacità di sviluppo, misurata dal *development capacity*: tale valore costituisce, infatti, il limite superiore per l'*ascendancy* (Allesina, 2004).

Il *development capacity* di un sistema si calcola pesando la *joint entropy* sulla base dell'indice TST:

$$C = TST \times H_{I,O}$$

$$C = \sum_{i=0}^{N+2}\sum_{j=0}^{N+2} t_{ij} \log\left(\frac{t_{ij}}{t_{..}}\right)$$

Per il caso studio in figura 2.1 si ha: $C = 6016{,}1$ bit.

Applicato nel contesto industriale, il *development capacity* rappresenta il potenziale massimo di sviluppo a disposizione di una rete di fornitura.

Inoltre, il *development capacity* si può suddividere in due componenti: l'*ascendancy*, che misura il livello di organizzazione dei flussi, e l'*overhead* (Φ), che indica la ridondanza e il grado di disorganizzazione dei flussi.

2.3.5. *Overhead* (Φ)

L'*overhead* indica la parte di *development capacity* che rimane disorganizzata, ovvero:

$$\Phi = C - A$$

$$\Phi = -\sum_{i=0}^{N+2}\sum_{j=0}^{N+2} t_{ij} \log\left(\frac{t_{ij}^2}{t_{i.}t_{.j}}\right)$$

Per il network in figura 2.1 si ha: $\Phi = 3011{,}4$ bit.

L'*overhead* è composto da quattro differenti contributi:

- *overhead in import* (Φ_I);
- *overhead in export* (Φ_E);
- *overhead in dissipation* (Φ_D);
- *redundancy* (R).

Le prime tre componenti sono collegate agli scambi che avvengono tra il sistema e l'esterno, mentre l'ultimo contributo dipende dalle sovrapposizioni dei percorsi all'interno del sistema (valori elevati di *redundancy* rispecchiano un'elevata presenza di percorsi paralleli).

Di solito i quattro indici sopra elencati vengono espressi come percentuale rispetto al *development capacity*: questo permette di confrontare, l'uno con l'altro, diversi network.

In tabella 2.4 sono riportati gli indici entropici introdotti finora (*ascendancy*, *capacity* e *overhead*) e il loro valore numerico con riferimento al network in figura 2.1.

Indici	**Valore**	**Percentuale**
$\mathrm{TST} = \sum_{i=0}^{N+2}\sum_{j=0}^{N+2} t_{ij} = t_{..}$	1573	
$C = \mathrm{TST} \times H_{\mathrm{I,O}} = \sum_{i=0}^{N+2}\sum_{j=0}^{N+2} t_{ij} \log\left(\frac{t_{ij}}{t_{..}}\right)$	6016,1	100,00%
$A = \mathrm{TST} \times \mathrm{AMI} = \sum_{i=0}^{N+2}\sum_{j=0}^{N+2} t_{ij} \log\left(\frac{t_{ij}t_{..}}{t_{i.}t_{.j}}\right)$	3004,7	49,94%
$\Phi_{\mathrm{I}} = -\sum_{j=1}^{N} t_{0,j} \log\left(\frac{t_{0,j}^2}{\sum_{i=1}^{N} t_{ij} \sum_{j=1}^{N} t_{0,j}}\right)$	689,2	11,46%
$\Phi_{\mathrm{E}} = -\sum_{j=1}^{N} t_{i,N+1} \log\left(\frac{t_{i,N+1}^2}{\sum_{j=1}^{N} t_{ij} \sum_{i=1}^{N} t_{i,N+1}}\right)$	344,8	5,73%
$\Phi_{\mathrm{D}} = -\sum_{j=1}^{N} t_{i,N+2} \log\left(\frac{t_{i,N+2}^2}{\sum_{j=1}^{N} t_{ij} \sum_{i=1}^{N} t_{i,N+2}}\right)$	404,6	6,72%
$R = -\sum_{i=1}^{N}\sum_{j=1}^{N} t_{ij} \log\left(\frac{t_{ij}^2}{\sum_{j=1}^{N} t_{ij} \sum_{i=1}^{N} t_{ij}}\right)$	1572,8	26,14%

Tabella 2.4 Indici entropici calcolati per la supply chain in figura 2.1 (formule tratte da Allesina *et al.*, 2010)

2.4.Analisi dei ricircoli

Il ricircolo di energia e di materia all'interno dei network ecologici è una delle caratteristiche più importanti di un ecosistema perché influenza il tempo di permanenza dei nutrienti (Herendeen, 1989), agisce da buffer (riserva) per le fluttuazioni nella fornitura di energia (Loreau, 1994), aumenta la stabilità (DeAngelis *et al*., 1989), e ha notevoli implicazioni sul funzionamento del sistema (Allesina, 2004).

Anche se in ambito ecologico la presenza di cicli trofici fu scoperta già nel 1948 (Hutchinson, 1948), si deve arrivare alla fine degli anni settanta per trovare un metodo che consente di quantificare l'ammontare effettivo dei ricircoli di materia e di energia negli ecosistemi. Tale metodologia, introdotta da Finn nel contesto generale dell'ENA (Finn, 1976), consente di ricavare il cosiddetto "indice di ricircolo di Finn" (*Finn's cycling index*, FCI), che fornisce una stima della percentuale dei flussi generati da cicli.

Uno dei vantaggi dell'indice di Finn risiede nella sua semplicità di elaborazione (richiede solo il calcolo di una matrice inversa). Inoltre, essendo un indice percentuale, è adimensionale e quindi permette agli ecologisti di comparare, direttamente tra loro, diversi ecosistemi.

Nel seguito della trattazione viene presentato l'indice di Finn, adattandolo però al contesto industriale, con riferimento al network in figura 2.1.

2.4.1.Matrice di Leontief e indice di Finn

Come precedentemente accennato, l'indice di Finn pone le sue radici nell'ambito dell'analisi dei network ecologici e, nello specifico, nell'analisi input/output. Il cuore di questa metodologia deriva da due modelli presi dall'economia: il primo fu sviluppato da Leontief (1963) al fine di stimare l'ammontare di materie prime necessarie per produrre una certa quantità di beni, mentre il secondo da Augostinovics (1981) che va in direzione opposta, stimando l'ammontare di output data una certa quantità di input (materiali e servizi). Introdotti nell'ecologia dall'ENA (Hannon, 1973), entrambi i metodi richiedono il calcolo di una matrice inversa (rispettivamente la matrice di Leontief e la matrice di Augostinovics).

Di seguito viene presentata la procedura che consente di ricavare prima la matrice di Leontief, quindi l'indice di Finn, sempre con riferimento al network industriale in figura 2.1.

Si parte dalla matrice degli scambi T e si normalizzano le sue colonne dividendo ogni coefficiente T_{ij} per il suo corrispondente flusso in entrata S_i, pari alla somma di tutti i beni che arrivano al nodo i:

$S_i = T_{.i} + I_i$

dove $T_{.i}$ rappresenta la somma dei coefficienti appartenenti alla i-esima colonna della matrice T. Ne risulta la cosiddetta matrice frazionaria dei flussi interni (*fractional inflow matrix*, G), nelle cui colonne si può leggere la composizione dei flussi in entrata verso ogni nodo: presa la colonna appartenente all'attore j, il valore nella riga i rappresenta la frazione del flusso totale entrante in j che proviene da i. La somma della colonna sarà esattamente uguale ad uno se l'attore j della supply chain viene rifornito di prodotti solo attraverso trasferimenti interni al sistema (dunque i flussi che arrivano in j partono solo da attori appartenenti al network), altrimenti sarà minore di uno se esiste un flusso in entrata verso j.

Moltiplicando la matrice G per se stessa (in modo da ottenere G^2), risulta una matrice i cui valori rappresentano la frazione del flusso totale che parte dall'attore individuato dalla riga i e arriva all'attore nella colonna j, attraverso un percorso che conta esattamente due step.

Nello stesso modo, G^3 individua la probabilità che i flussi arrivino nella colonna j attraverso tutti i percorsi che contano esattamente tre step, e così via per ogni potenza di G.

Se il sistema contiene cicli, la matrice G^n tenderà a 0 con $n \to \infty$: i coefficienti della matrice diventano infinitamente piccoli dopo soli pochi step, ma non si annullano mai del tutto a causa dei ricircoli.

Per ricavare la matrice di Leontief si deve ricorrere alla seguente serie geometrica:

$\sum_{n=0}^{\infty} q^n = 1 + q + q^2 + ... = \frac{1}{1-q}$, valida se $-1 < q < 1$.

Dunque, nel caso in cui $0 \le G_{ij} \le 1$, è possibile dimostrare (Higashi *et al*., 1991) che:

$$\sum_{n=0}^{\infty} G^n = G^0 + G + G^2 + ... = [I - G]^{-1} = L \qquad (1)$$

dove G^0 è uguale alla matrice identica I, mentre L è la matrice inversa di Leontief.

In tabella 2.5 si riportano le matrici G e L calcolate per il network in figura 2.1.

In accordo con Higashi *et al*. (1991), ogni coefficiente della matrice di Leontief rappresenta il numero medio di volte che la frazione di un bene[11], collocato presso l'attore in colonna j, arriverà all'attore nella rispettiva riga i.

[11] Higashi *et al*. (1991), per enfatizzare il concetto, parlano di particella di energia/materia.

Matrice G

	S1	S2	S3	M	C1	C2	R
S1	0	0	0	0,3851	0	0	0
S2	0	0	0,288	0,2126	0	0	0
S3	0	0,4348	0	0,342	0	0	0
M	0	0	0	0	1	1	0
C1	0	0	0	0	0	0	0
C2	0	0	0	0	0	0	1
R	0	0	0	0,0603	0	0	0

Matrice L

	S1	S2	S3	M	C1	C2	R
S1	1	0	0	0,4098	0,4098	0,4098	0,4098
S2	0	1,1431	0,3292	0,3785	0,3785	0,3785	0,3785
S3	0	0,497	1,1431	0,5285	0,5285	0,5285	0,5285
M	0	0	0	1,0642	1,0642	1,0642	1,0642
C1	0	0	0	0	1	0	0
C2	0	0	0	0,0642	0,0642	1,0642	1,0642
R	0	0	0	0,0642	0,0642	0,0642	1,0642

Tabella 2.5 Matrice frazionaria dei flussi interni (G) e matrice di Leontief (L)

Come si può vedere da (1), il numero medio di volte che un prodotto, una volta entrato nel nell'i-esimo compartimento, visiterà l'i-esimo compartimento (gli elementi sulla diagonale della matrice L) è almeno uno. Coefficienti maggiori dell'unità indicano che il compartimento (attore) fa parte di un ricircolo.

A questo punto è possibile calcolare l'indice FCI (Finn, 1976) che consente di valutare l'ammontare di materiale che ricircola all'interno del network.

La formula deriva proprio dalla matrice inversa di Leontief:

$$FCI = \sum_{i=1}^{n} \frac{S_i}{TST} \frac{l_{ii} - 1}{l_{ii}}$$

dove l_{ii} è il coefficiente i-esimo lungo la diagonale della matrice di Leontief, e S_i è il flusso in entrata nell'i-esimo compartimento. L'indice TST (total system throughput) è già stato descritto nel sottoparagrafo 2.3.1: rappresenta la somma di tutti i flussi che concorrono nell'ecosistema e si calcola sommando i coefficienti di tutte le matrici: $T_{..} + Z_{.} + E_{.} + D_{.}$.

Per il caso oggetto di studio in figura 2.1, l'indice di Finn è pari a: $FCI = 4{,}95\%$.

Significa che circa il 5% dei flussi totali è generato mediante ricircoli o, in altre parole, che circa il 5% del materiale ricircola all'interno della supply chain.

2.4.2.Tipi di ricircolo e loro probabilità

L'indice di Finn, calcolato per un supply network, rappresenta la frazione dei flussi logistici generati di ricircoli. Essendo adimensionale, permette di confrontare reti industriali diverse. Tuttavia, se usato da solo, l'FCI si rivela di scarsa utilità: contrariamente a quello che accade negli ecosistemi, all'interno di una rete di fornitura esistono molteplici tipologie di ricircolo, aventi origini, cause ed effetti diversi.

Come si può intuire, qualsiasi flusso logistico di ricircolo complica la gestione di una supply chain nel suo complesso. Tuttavia, se si tratta di un ricircolo che, ad esempio, permette di recuperare valore da un prodotto invenduto destinato allo smaltito in discarica, allora l'aumento di complessità del network può essere controbilanciato dal ritorno economico che ne deriva, o comunque giustificato da aspetti normativi e/o etici. D'altro canto, se il flusso logistico di ricircolo è causato dal ritorno di prodotti che, ad esempio, non soddisfano le specifiche richieste dal cliente, si avrà un aumento ingiustificato della complessità della supply chain.

Da qui la necessità di discriminare i flussi logistici di ricircolo: una loro possibile classificazione è stata proposta nel capitolo 1 della tesi.

La procedura inizia dall'individuazione dei ricircoli in un supply network e della loro natura:

- riuso/rivendita/redistribuzione diretti.
- pulizia e reimballaggio (*clean and repackage*);
- riparazione (*repair*);
- revisione (*refurbishing*);
- rilavorazione (*remanufacturing*);
- recupero parti (*parts retrieval*);
- riciclaggio (*recycling*);
- incenerimento o smaltimento in discarica.
- conto-lavoro.

Nel network in figura 2.1 sono presenti due ricircoli:

a) S3 → S2 → S3: si tratta di un ricircolo dovuto ad un conto-lavoro;
b) C2 → R → M → C2: concerne il riciclaggio di materiale.

Il passo successivo consiste nel calcolare la probabilità di ogni ciclo, che equivale al prodotto dei coefficienti della matrice G che appartengono agli archi del ricircolo stesso.

Tornando al caso studio, ricircolo (a), ha una probabilità pari a:

$$P_a = G_{S3\to S2} \cdot G_{S2\to S3} = 0{,}4348 \cdot 0{,}288 = 0{,}1252 = 12{,}52\%$$

La probabilità di (b) è, invece:

$$P_b = G_{C2\to R} \cdot G_{R\to M} \cdot G_{M\to C2} = 1 \cdot 0{,}0603 \cdot 1 = 0{,}0603 = 6{,}03\%$$

Significa che un prodotto ha una probabilità del 12,52% di partecipare al ricircolo (a) e del 6,03% di eseguire nel ricircolo (b).

A questo punto è possibile calcolare la frazione di probabilità di ogni ciclo sull'ammontare totale dei ricircoli:

a) $$\frac{P_a}{P_a + P_b} = \frac{0{,}1252}{0{,}1252 + 0{,}0603} = 0{,}6748 = 67{,}48\%$$

b) $$\frac{P_b}{P_a + P_b} = \frac{0{,}0603}{0{,}1252 + 0{,}0603} = 0{,}3252 = 32{,}52\%$$

Su cento prodotti che partecipano ai ricircoli, si può affermare che circa 67 appartengono al ricircolo (a) e circa 33 a (b). Questa considerazione risulta utile in particolar modo se è presente una logistica inversa che recupera i prodotti e li fa riconvergere presso un'azienda dove verranno riprocessati: sapere che, ad esempio, su cento prodotti 67 dovranno presumibilmente essere riciclati, mentre 33 probabilmente riparati, può essere di aiuto nel programmare, in anticipo, la capacità delle operations.

L'indice di Finn, se integrato con le informazioni appena ricavate, diventa un indicatore esaustivo, capace di fornire una visione globale sull'ammontare dei flussi logistici di ricircolo e, allo stesso tempo, di approfondire il fenomeno considerando le caratteristiche individuali di ogni ricircolo appartenente allo specifico network.

Inoltre, l'FCI si colloca nell'ambito generale di analisi della distribuzione e dell'organizzazione dei flussi all'interno di un supply network: l'indice di Finn è uno dei tasselli di un puzzle che, per essere completo, necessita degli altri pezzi, gli indici entropici. Per non incappare in valutazioni fuorvianti (spesso causate dall'insufficienza di informazioni/dati) è dunque opportuno esaminare il comportamento di tutti gli indici presentati in questo capitolo, soprattutto quando si vuole comparare tra loro supply chain diverse.

CAPITOLO 3

Supply network analysis supportata da software

3.1.Modellazione dei network istanza test

In questo capitolo verranno modellate, grazie a quanto ricavato dall'analisi della letteratura scientifica, delle istanze test di supply network: nello specifico, le strutture dei flussi logistici che verranno nel seguito della trattazione proposte si rifanno a quelle descritte nel lavoro di Blackburn *et al.* (2004).

Si vuole innanzitutto presentare ed esaminare una prima configurazione di una ipotetica rete di fornitura (istanza test 1) e proporre un'alternativa (istanza test 2) che prevede la riprogettazione della logistica inversa. Il caso verrà contestualizzato riportando argomentazioni e supposizioni riprese dalla letteratura allo scopo di rispecchiare verosimilmente le esigenze che possono indurre le aziende reali a rivedere la loro strategia di supply chain management.

Successivamente verranno confrontati tra loro i due network presentati sulla base degli indici di complessità introdotti nel secondo capitolo, prestando particolare attenzione al comportamento dell'indice di Finn.

Infine si proporranno altre due configurazioni (istanza test 3 ed istanza test 4) complementari alle istanze test precedentemente introdotte, con l'aggiunta di alcuni flussi relativi al riciclaggio di materiale.

L'intera analisi verrà supportata da un software sviluppato ad hoc in MATLAB, ambiente per il calcolo numerico e l'analisi statistica che comprende anche l'omonimo linguaggio di programmazione.

Prima di passare alla presentazione e allo studio del caso fittizio, si riportano qui di seguito le strutture dei flussi logistici inversi a cui fanno riferimento tutte le istanze test: come precedentemente accennato si tratta di configurazioni riprese direttamente dalla letteratura scientifica, riscontrabili in molte realtà che implementano il processo di recupero dei resi da parte dei clienti. Va inoltre ribadito che il recupero e la raccolta dei resi, insieme alle attività di trasporto, ricezione e smistamento, rientrano nell'ambito della logistica inversa (o di ritorno) (Blumberg, 2005).

Blackburn *et al.* (2004) individuano due tipologie opposte di supply chain inverse, distinguendo quelle efficienti da quelle reattive, a seconda dell'obiettivo perseguito. I network efficienti mirano a ridurre al massimo i costi della logistica inversa spesso a scapito del consumatore, costretto ad attendere tempi lunghi per riavere in mano il prodotto reso, ad esempio, in quanto difettoso. Dalla parte opposta, invece, i network reattivi sono in grado di fornire una risposta rapida ai clienti, velocizzando le attività di recupero, ripristino e riconsegna del prodotto al cliente, ma pagando di più in termini di efficienza e costi.

In accordo con quanto affermato da Blackburn *et al.* (2004), la progettazione della supply chain inversa si basa su un *trade off* tra velocità e costi. Non esiste, tuttavia, una soluzione ottimale che sia univoca. Secondo gli autori sopra citati essa dipende dalle caratteristiche del prodotto: nello specifico, se si tratta di un prodotto *time-sensitive*, dunque caratterizzato da un elevato tasso di perdita di valore nel tempo, la logistica di ritorno dovrà essere il più possibile reattiva; nel caso opposto, in cui il valore del prodotto non è pesantemente influenzato dal tempo, potrà puntare all'efficienza.

Per quanto riguarda la struttura del network, la principale differenza tra le due tipologie di supply chain inverse appena presentate, risiede nella collocazione delle attività di test e valutazione dei prodotti di ritorno, al fine di determinare lo stato (e quindi l'opzione di recupero più idonea) per ogni articolo. Se l'obiettivo da perseguire è l'efficienza, allora tali attività dovrebbero essere centralizzate (figura 3.1) in modo da sfruttare le economie di scala sia nel trasporto dei prodotti che nel loro trattamento; se si punta alla reattività del network, la valutazione dello stato del prodotto dovrebbe essere il più possibile decentralizzata (figura 3.2) per minimizzare i ritardi lungo il processo di ritorno dei resi.

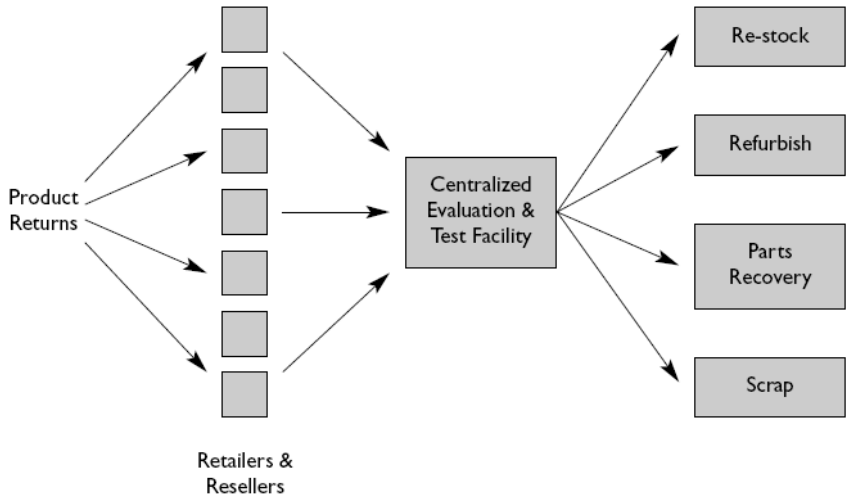

Figura 3.1 Supply chain inversa: modello centralizzato
(tratto da Blackburn *et al.*, 2004)

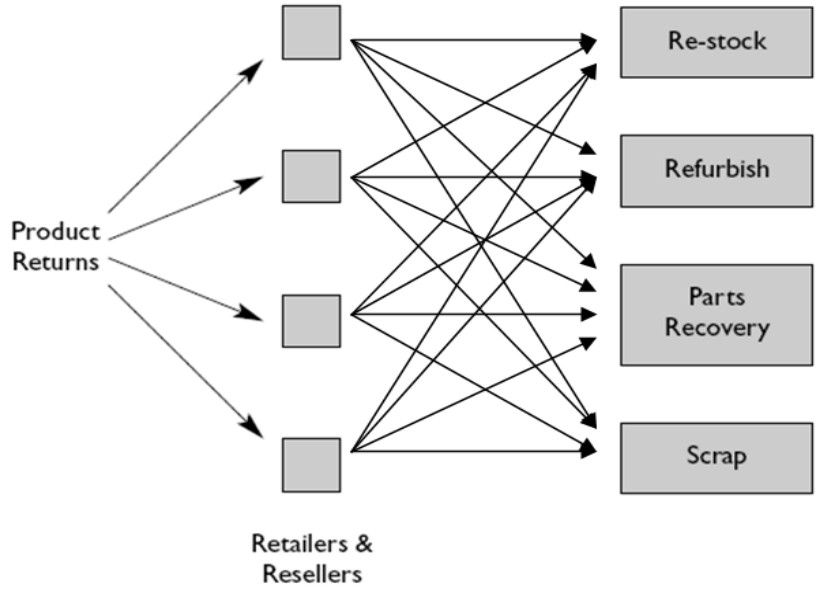

Figura 3.2 Supply chain inversa: modello decentralizzato
(tratto da Blackburn *et al.*, 2004)

I modelli centralizzato e decentralizzato sono ovviamente due soluzioni agli antipodi: esistono infatti delle configurazioni intermedie nelle quali le attività di valutazione vengono parzialmente decentralizzate così da ottenere un buon bilanciamento tra costi e velocità di risposta ai clienti.

3.2.Istanza test 1: network efficiente

L'istanza test 1 oggetto di analisi riguarda un'ipotetica supply chain che commercializza articoli elettronici di largo consumo come Personal Computer e apparecchi per la telecomunicazione: si tratta, quindi, di prodotti *time-sensitive*[12].

La supply chain implementa una logistica di ritorno che si occupa innanzitutto di recuperare i prodotti restituiti dai clienti agli stessi punti vendita in cui erano stati precedentemente acquistati, quindi di processarli secondo l'opzione di recupero più idonea ed infine di riconsegnarli ai *retailer*.

I resi accettati dai negozianti ricoprono un ampio ventaglio di casistiche: dagli articoli difettosi in garanzia che necessitano di un intervento di ripristino, a quelli danneggiati che devono essere riparati (anche se non rientrano in garanzia), fino a quelli in perfette

[12] Blackburn *et al.* (2004) riportano il tasso di perdita di valore per i PC: si tratta di una percentuale superiore all'1% a settimana, che aumenta man mano che il prodotto si avvicina alla fine della sua vita utile.

condizioni, ma che vanno sostituiti in quanto contemplato da politiche commerciali del tipo “soddisfatti o rimborsati”.

I *retailer*, pur costituendo i punti iniziali (e finali) dei flussi della logistica inversa, non svolgono nessun ruolo attivo nel processo di ritorno: essi sono semplicemente dei luoghi di raccolta in cui confluiscono i prodotti restituiti dai clienti. I resi vengono poi inviati in un unico centro per essere testati e valutati. Tuttavia, per ridurre i costi di trasporto, spesso i *retailer* ritardano le spedizioni fintantoché non hanno accumulato un lotto considerevole di prodotti.

Nel centro test-valutazione convergono quindi i resi provenienti da tutti i punti vendita sparsi nel network. È qui che si decide il futuro di ogni prodotto: se è in ottimo stato, prima lo si sottopone ad una veloce pulizia e al reimballaggio nel posto, quindi viene spedito al centro di distribuzione (CEDI) da dove partono i camion che riforniscono i vari *retailer;* se, nel caso opposto, è completamente usurato, viene scartato direttamente in loco; se invece è ripristinabile, una volta scelta l’opzione di recupero più idonea tra la riparazione e la rilavorazione, viene inviato alla struttura corrispondente per essere processato e da qui spedito al solito CEDI.

Nel seguente paragrafo viene mappata la supply chain dell’istanza test 1 appena descritta, presentando nel contempo il software sviluppato per l’analisi dei network.

3.2.1.Mappatura dell’istanza test mediante software

Per poter studiare una supply chain utilizzando gli strumenti appartenenti all’analisi del network è necessario disporre della matrice estesa dei trasferimenti T^* (vedi paragrafo 2.2), i cui dati identificano una certa tipologia di flussi di scambio.

Requisito essenziale è tale matrice sia in stato di equilibrio (*steady state*).

In tabella 3.1 è riportata la matrice T^* per l’istanza test 1, contenente i flussi di materiale e di beni misurati in tonnellate/anno.

Da MATLAB si avvia il programma[13] che visualizza la schermata riportata in figura 3.3: da qui è possibile caricare il file contenente la matrice estesa T^* del network oggetto di studio (in questo caso si carica il foglio di calcolo contenete la tabella 3.1).

[13] Una volta entrati nell’ambiente MATLAB, per avviare il software basta lanciare in esecuzione il file “finestraPercorsoFile.m”.

Comparti	Imp.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	Exp.	Diss.
Imp.		300	667	222	114	80			668	363																	
1							285																				15
2								648																			19
3								222																			
4								109																			5
5								72																			8
6											275																10
7											793			240													18
8												662	157														
9														354													9
10														1056													12
11														645													17
12									151																		6
13															1723	572											
14																	1723										
15																	572										
16																		1050	282	924	337	149					
17																							254			796	
18																							62			220	
19																							221			703	
20																							77			260	
21																							32			117	
22																	115							273	136		122
23																	245										28
24																	87										49
Exp.																											
Diss.																											

Tabella 3.1 Matrice estesa dei trasferimenti T^* dell'istanza test 1

Figura 3.3 Finestra da cui caricare la matrice estesa T^*

Premendo su "ok", il software elabora ed interpreta i dati della matrice T^* e propone un primo risultato grafico. Si tratta di una rappresentazione del network mediante grafo orientato.

Nonostante tale strumento sia previsto anche nell'ENA (vedi capitolo 2, paragrafo 2.1), per mappare una supply chain industriale si devono seguire criteri ben diversi da quelli richiesti per un ecosistema. La modellazione convenzionale di una supply chain, infatti, segue il flusso dei beni lungo la catena diretta, partendo dai fornitori fino ad arrivare ai clienti finali.

Per ottenere un grafico del genere, il programma implementa una procedura ricorsiva che, esplorando tutti i nodi in successione, individua il percorso più lungo: il numero di attori che compone tale percorso equivale al numero di livelli in cui è suddiviso il grafo. Gli altri attori vengono poi inseriti, uno alla volta a ritroso nel grafico, partendo

dall'ultimo livello (quello dei clienti) e risalendo verso il primo livello (i fornitori), fintantoché presentano dei successori.

Disporre del modello di un network significa avere uno strumento visivo in grado di comunicare in maniera efficace informazioni riguardanti anche la complessità della rete stessa, che tuttavia non emergono all'istante guardando la matrice estesa. Prima tra tutte, la distribuzione dei collegamenti tra i vari attori all'interno del network: è immediato vedere, ad esempio, se la rete di fornitura è sbilanciata, nel numero di attori, dal lato dei fornitori o dei clienti, oppure se i flussi sono organizzati lungo un percorso preferenziale o se esistono dei flussi logistici di ricircolo, oppure se ci sono attori spaiati, etc.

D'altro canto, mappare una rete industriale che presenta centinaia di nodi partendo dalla matrice estesa, potrebbe rivelarsi un'impresa ardua e lunga. Considerando che, nelle analisi di complessità dei network, è normale avere a che fare con un numero elevato di attori, si può trarre il massimo vantaggio usando un software in grado di restituire una prima rappresentazione automatica e istantanea dell'intera rete.

In figura 3.4 è riportato il grafo ottenuto dalla matrice estesa dell'istanza test 1.

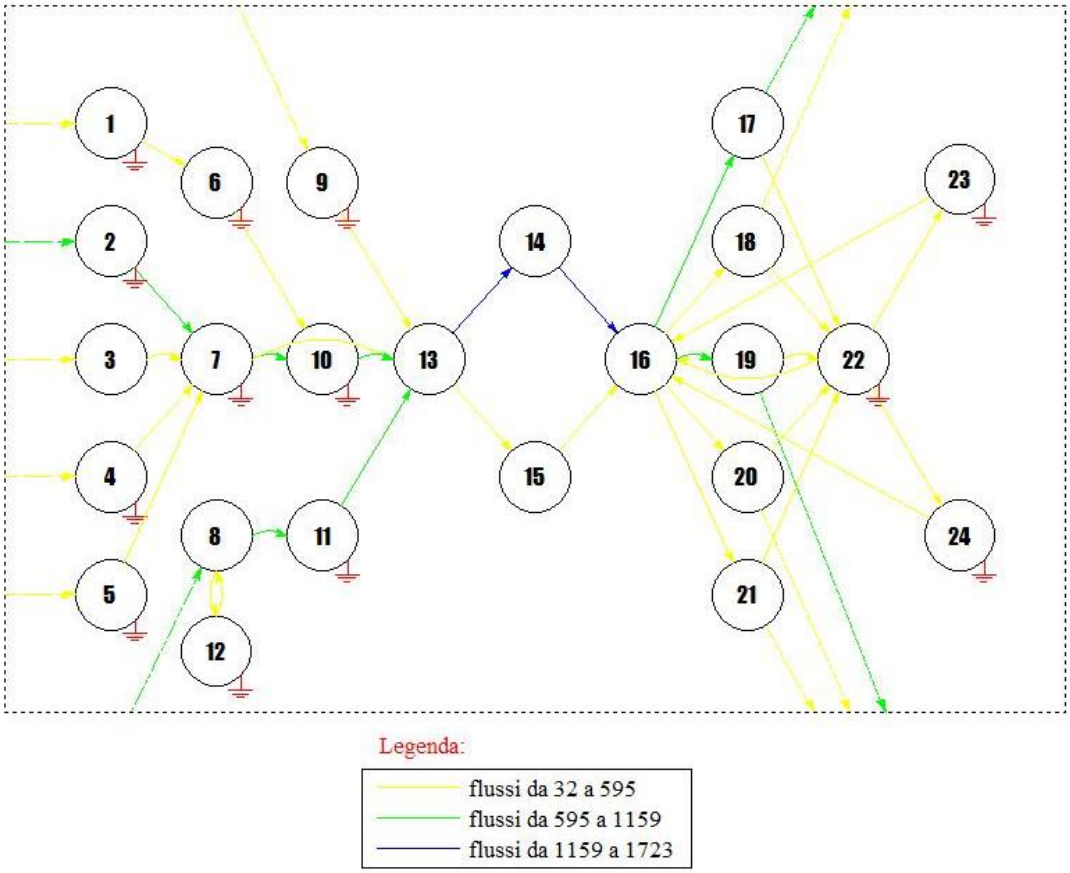

Figura 3.4 Rappresentazione del network istanza test 1

Come si può notare, la supply chain conta dodici fornitori di materie prime o di componenti, collegati tra loro da flussi di scambio (tonnellate di materiale/anno) convergenti in un CEDI (nodo 13). Da qui partono i rifornimenti di materiale diretti a due OEM che assemblano i prodotti finali e li inviano al secondo CEDI (nodo 16). Quest'ultimo rifornisce, a sua volta, cinque *retailer*. Da questo punto in poi subentra la logistica inversa: i resi dei vari punti vendita vengono spediti nel centro test e valutazione (nodo 22), quindi possono essere inviati alle strutture di riparazione (nodo 23) o di rilavorazione (nodo 24) e infine ritornare al secondo CEDI. Dal lato dei fornitori si notano due flussi preferenziali (in verde) e un ricircolo tra il nodo 8 e il nodo 12. Dal lato dei clienti si distinguono, invece, tutti i ricircoli passanti per il centro test e valutazione.

3.2.1.1.Modificare il grafo

Si può intervenire sul grafo proposto mediante l'interfaccia riportata in figura 3.5.
Innanzitutto, se il risultato grafico ottenuto non è per nulla soddisfacente (potrebbe accadere nel caso in cui la matrice data in input sia parecchio disordinata), è presente un pulsante con il quale si richiama una funzione che interviene sulla matrice, la riordina secondo determinati criteri[14] e ripropone un nuovo grafo.

Figura 3.5 Finestra che consente di intervenire sul grafo

[14] Per approfondire questo aspetto si rimanda al codice allegato alla fine della tesi, che riporta tutte le funzioni implementate dal software con i relativi commenti che ne descrivono il funzionamento e le logiche sottostanti: in questo caso si veda la funzione "ordinaMatrici.m".

Nella schermata in figura 3.5, attraverso il pulsante “sposta”, è possibile ricollocare, all’interno del grafo, il nodo selezionato. È possibile intervenire anche sugli spazi tra ogni livello, con l’effetto di dilatare o comprimere il grafo.
Questi accorgimenti consentono di migliorarne l’aspetto finale del grafo.
Premendo il tasto “conferma” si passa alle successive schermate.

3.2.1.2.Inserire informazioni nel grafo

La schermata riportata in figura 3.6 consente di visualizzare (o nascondere) i flussi degli archi nel grafo e di impostare le dimensioni del carattere.
Inoltre, selezionando un nodo dal menù a tendina, si visualizzano i valori dei rispettivi flussi in uscita e in entrata, quelli verso gli altri nodi e le dissipazioni. Un’icona permette di sapere se il nodo si trova in *steady state* (se per il nodo non sussiste il bilancio delle masse, l’icona appare barrata).

Figura 3.6 Schermata mediante cui inserire le informazioni nel grafo

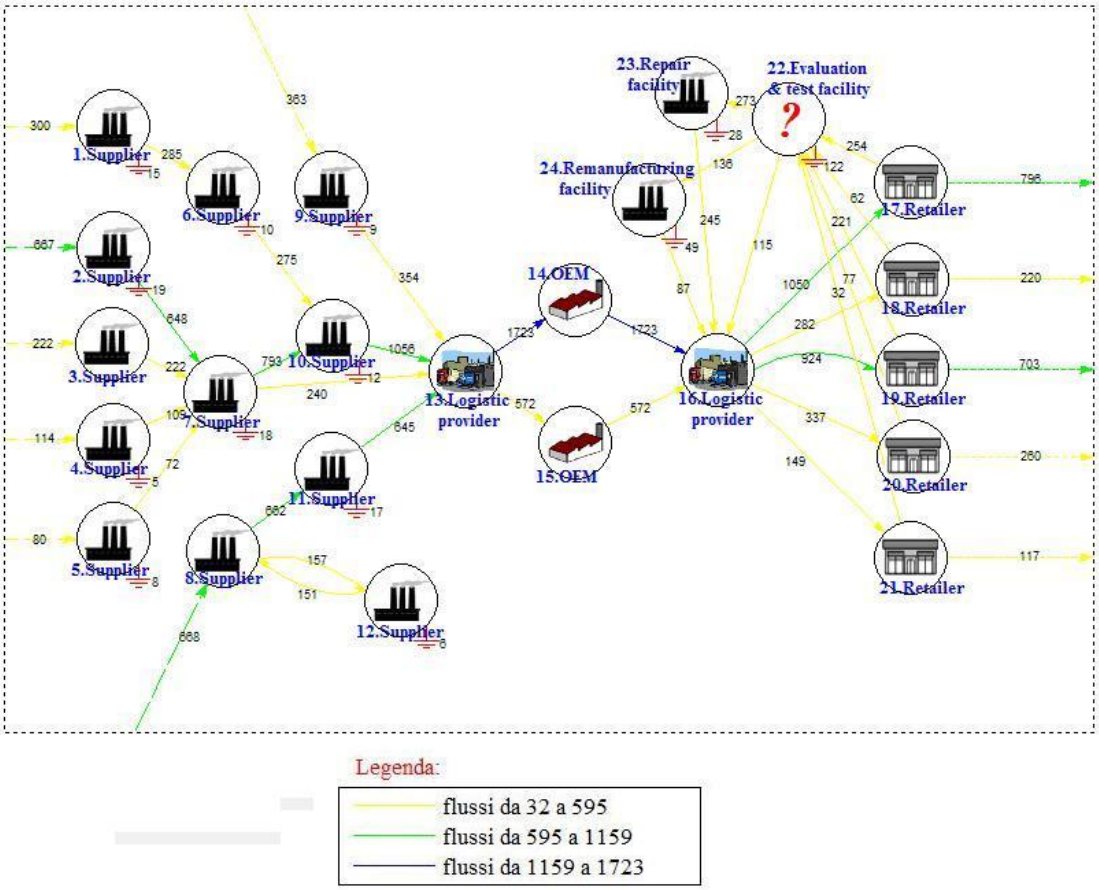

Figura 3.7 Possibile grafo finale del network istanza test 1

Sempre dalla stessa finestra si possono inserire nel grafo delle informazioni riguardanti il nodo selezionato: dopo aver scelto (o inserito) la categoria a cui appartiene, che corrisponde al ruolo dell'attore all'interno della supply chain (fornitore di materie prime, piuttosto che un *provider* di servizi logistici o un retailer, etc.), si possono inserire nel grafo l'icona e l'etichetta relative.
Una possibile mappatura finale per l'istanza test 1 è riportata in figura 3.7.

3.2.2.Individuazione dei ricircoli e calcolo degli indici

Il software, dopo aver individuato tutti i flussi logistici di ricircolo presenti nel network[15], li riporta nel menù a tendina della finestra in figura 3.8.
A questo punto, all'utente non resta altro che specificare la natura di ogni ricircolo scegliendo tra le categorie individuate nel primo capitolo della tesi (figura 3.9).

[15] La procedura ricorsiva che consente di trovare tutti i ricircoli è implementata nella funzione "trovaCicli.m" in allegato alla tesi.

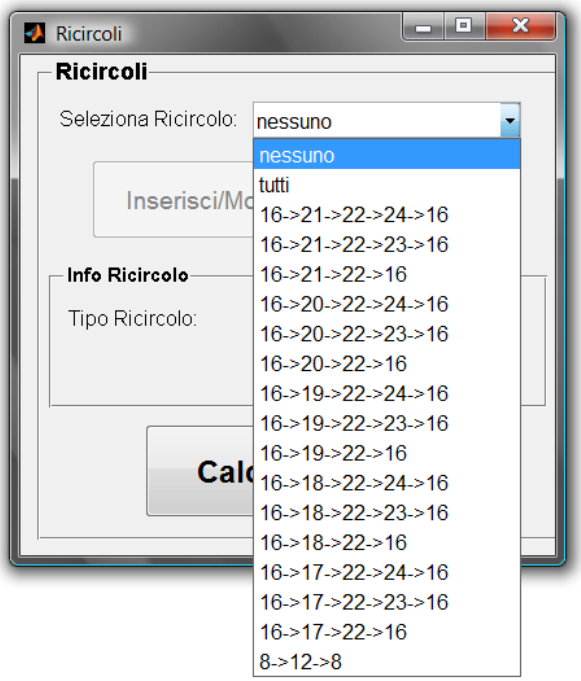

Figura 3.8 Schermata con i ricircoli presenti nel network

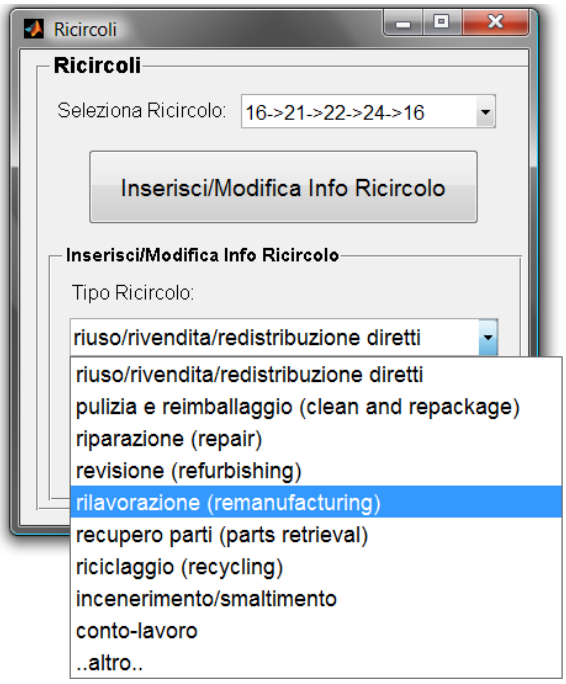

Figura 3.9 Selezione del tipo di ricircolo

Nell'istanza test 1 sono presenti sedici ricircoli (figura 3.8), di cui cinque prevedono l'attività di riparazione, cinque la rilavorazione, cinque la pulizia e il reimballaggio ed uno riguarda un conto-lavoro.

Premendo il pulsante "calcola indici" compare un'ultima finestra nella quale si selezionano gli output che si vogliono ottenere dall'analisi (figura 3.10).

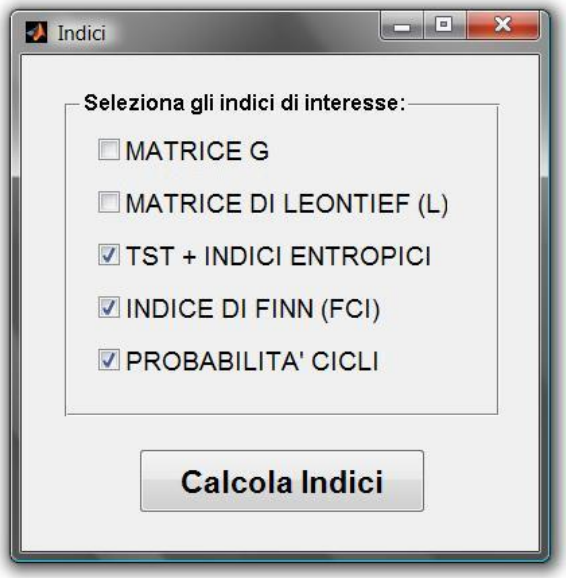

Figura 3.10 Finestra con gli output da calcolare

3.2.3.Analisi dei risultati

Una volta elaborati i risultati, il software li riporta automaticamente in un foglio Excel. Inoltre, nel caso lo richieda l'utente, insieme al calcolo degli indici entropici il programma restituisce in output due diagrammi per una lettura grafica immediata delle percentuali di tali indici.

Per l'istanza test 1 si riportano i valori dei principali indici e i risultati dell'analisi dei ricircoli nelle tabelle 3.2, e i diagrammi in figura 3.11.

> INDICI:	TST	AMI	C	A	ϕ	ϕ I	ϕ E	ϕ D	R
VALORE:	19331	3,1473	97822	60840	36982	6227	5017	2015	23722
PERCENTUALE:	-	-	100	62,19	37,81	6,37	5,13	2,06	24,25

> FCI: 0,0471

> TIPI DI RICIRCOLO	PROB	PROB%
riuso/rivendita/redistribuzione diretti	0	0
pulizia e reimballaggio (clean and repackage)	0,04194	12,07
riparazione (repair)	0,08935	25,72
revisione (refurbishing)	0	0
rilavorazione (remanufacturing)	0,03173	9,13
recupero parti (parts retrieval)	0	0
riciclaggio (recycling)	0	0
incenerimento/smaltimento	0	0
conto-lavoro	0,18437	53,07

Tabelle 3.2 Valori dei principali indici e probabilità di ogni tipologia di ricircolo, calcolati per il network l'istanza test 1

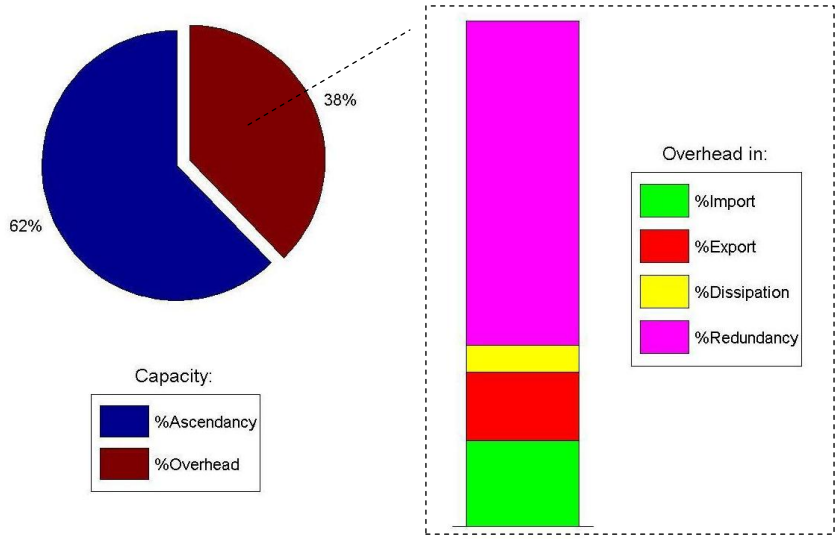

Figura 3.11 Diagrammi degli indici entropici per il network istanza test 1

L'indice di Finn, pari a 4,71%, consente di stimare l'ammontare di beni che ricircolano nel network istanza test 1. Inoltre, la frazione di probabilità calcolata per ogni flusso logistico di ricircolo dimostra che più del 50% del materiale oggetto di ricircolo partecipa al conto-lavoro, mentre i cinque ricircoli che prevedono la rilavorazione (*remanufacturing*) dei resi non raggiungono insieme il 10%. L'*overhead* (Φ) e l'*ascendancy* (A) rappresentano rispettivamente circa il 38% e il 62%. Si tratta di un buon compromesso: network eccessivamente organizzati, dunque caratterizzati da elevati rapporti di A/Φ, si rivelerebbero troppo fragili a fronte di imprevisti o di cambiamenti improvvisi che provocassero l'interruzione di qualche flusso; d'altro canto, network troppo complessi, con elevati rapporti di Φ/A, sono contraddistinti da un'eccessiva presenza di flussi ridondanti che, oltre a ripercuotersi negativamente sui costi, possono causare la perdita di controllo del network stesso (Allesina *et al.*, 2010). Infine, dai diagrammi si vede che la componente dell'*overhead* che pesa maggiormente è proprio il *redundancy*, come ci si poteva aspettare osservando il grafo, vista la presenza di numerosi ricircoli e di percorsi paralleli.

3.3.Istanza test 2: network reattivo

La struttura della supply chain inversa dell'istanza test 1 risponde alle caratteristiche dei network efficienti: sfrutta le economie di scala ma costringe i clienti a lunghe attese con ovvie ripercussioni sull'immagine aziendale.

Dunque, in questo caso ipotetico, il principale motivo che spinge il management a rivedere la strategia di logistica inversa è strettamente collegato alla necessità di orientarsi verso modelli di servizio che pongono il cliente al centro dell'attenzione. Essendo inoltre commercializzati articoli *time-sensitive*, il risparmio che si ottiene avendo centralizzato l'attività di test e valutazione potrebbe essere vanificato dalla perdita di valore nel tempo subita dai prodotti stessi.

Per tutte queste ragioni, la riprogettazione del network deve tendere ad un modello decentralizzato, in cui l'attività di valutazione dei resi viene il più possibile anticipata già nei *retailer*, accelerando in tal modo i tempi richiesti per il processo di recupero dei prodotti.

Ovviamente, i punti vendita devono essere in grado di determinare lo stato dei resi, operazione che richiede un'attrezzatura specifica e/o un personale qualificato. In tal senso si perdono i vantaggi derivanti delle economie di scala: una soluzione di compromesso per non perdere troppo in efficienza, potrebbe essere una parziale decentralizzazione delle attività di valutazione. Ciò si può ottenere delegando ai *retailer* il compito di individuare quando un prodotto si trova in una delle due condizioni estreme: si tratta di capire, mediante semplici test che non richiedono elevati investimenti in attrezzature/formazione, se un articolo è proprio irrecuperabile, quindi da scartare nel posto, oppure se è come nuovo, quindi da rimettere subito in vendita (previa veloce pulizia e reimballaggio eseguiti in loco). Se il reso non appartiene ad una delle due categorie, viene spedito in uno o più centri specializzati per valutarne l'opzione di recupero: in questo modo non si perdono tutti i vantaggi derivanti dalle economie di scala.

L'istanza test 2 analizzata nel seguito della trattazione, presenta un modello completamente decentralizzato in cui tutte le attività di valutazione sopra descritte vengono effettuate presso ogni punto vendita. Questa soluzione, come precedentemente accennato, crea le condizioni per garantire risposte rapide ai consumatori e per soddisfare le loro crescenti aspettative sul livello di servizio fornito dalle aziende.

Punto di partenza della network analisi è sempre la matrice estesa T^*, contenente i flussi di beni misurati in tonnellate/anno (tabella 3.3).

In figura 3.12 è riportata la modellazione del network dell'istanza test 1 ottenuta con il software presentato nei precedenti paragrafi.

Comparti	Imp.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	Exp.	Diss.
Imp.		300	667	222	114	80			668	363																
1							285																			15
2								648																		19
3								222																		
4								109																		5
5								72																		8
6											275															10
7											793			240												18
8												662	157													
9														354												9
10														1056												12
11														645												17
12									151																	6
13															1723	572										
14																	1723									
15																	572									
16																		1003	272	888	324	140				
17																							108	57	796	42
18																							28	11	220	13
19																							96	46	703	43
20																							30	17	260	17
21																							11	5	117	7
22																	245									28
23																	87									49
Exp.																										
Diss.																										

Tabella 3.3 Matrice estesa dei trasferimenti T^* dell'istanza test 2

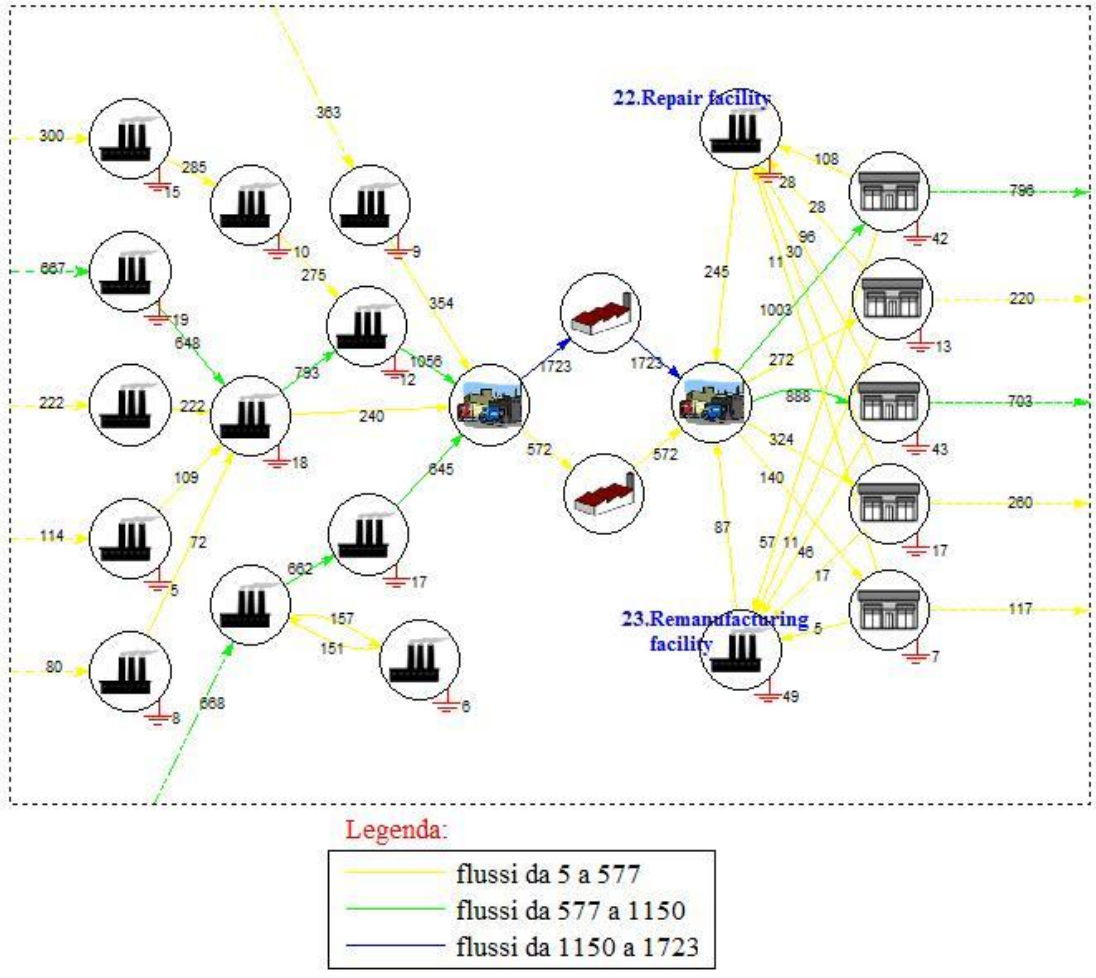

Figura 3.12 Grafo del network istanza test 2

Come si vede dal grafo, ogni *retailer* è direttamente collegato sia alla struttura in cui si effettuano le riparazioni, sia in quella per le rilavorazioni, senza passare per il centro test-valutazione. Questo è possibile perché ogni negoziante è in grado di determinare in modo autonomo lo stato dei resi e quindi di scegliere la strada più idonea da intraprendere: se il prodotto è come nuovo viene rimesso in vendita dopo aver tolto eventuali imperfezioni superficiali e averlo reimballato nel posto; se l'articolo è irrecuperabile, si elimina in loco; se deve essere riparato o rilavorato si manda nella rispettiva struttura. Nei primi due casi i beni non devono essere spediti da alcuna parte. Nel terzo caso i ricircoli dovuti alle attività di riparazione e rilavorazione, pur riguardando le medesime quantità di prodotti che girano nella configurazione dell'istanza test 1, bypassano il centro test-valutazione: il fatto che si coinvolgono meno attori si traduce in flussi di ricircolo più corti.

È da notare che si è passati da sedici ricircoli presenti nel network dell'istanza test 1, ad undici dell'istanza test 2, avendo di fatto "eliminato" la tipologia che prevedeva le attività di pulizia e reimballaggio.

In definitiva, per tutti i motivi illustrati, in questa seconda configurazione i ricircoli hanno un peso meno rilevante che, in linea teorica, deve rispecchiarsi in un minore indice di Finn e in una diminuzione di *redundancy*. Inoltre, sempre per le stesse ragioni, ci si aspetta una diminuzione dell'indice TST, essendo pari alla somma di tutti i flussi interni. Guardando il grafo in figura 3.12 ci si accorge, tuttavia, che la logistica di ritorno è diventata più caotica e che i flussi sono meno organizzati: questo probabilmente si rifletterà in un peggioramento dell'AMI.

Nelle tabelle 3.4 si riportano i valori dei principali indici e il risultato dell'analisi dei ricircoli per il network istanza test 1. Nelle tabelle 3.5 si possono leggere gli scostamenti degli indici passando dalla configurazione dell'istanza test 1 a quella dell'istanza test 2: nel confronto tra network è utile osservare i valori espressi come percentuale rispetto la capacità del sistema.

Come ci si aspettava, l'indice di ricircolo di Finn è diminuito. Anche le previsioni sugli indici entropici risultano corrette. Nel complesso i due network risultano avere proporzioni molto simili di A/C (entrambe intorno al 62%) e di conseguenza anche di Φ/C. Infatti, anche se passando dall'istanza test 1 all'istanza test 2 l'*ascendancy* è diminuito in valore assoluto, si hanno circa gli stessi rapporti di A/C per entrambe le configurazioni, essendo calato il *development capacity* di una percentuale simile all'*ascendancy*.

> INDICI:	TST	AMI	C	A	ϕ	ϕ I	ϕ E	ϕ D	R
VALORE:	18455	3,0781	91942	56806	35136	6227	4888	2510	21512
PERCENTUALE:	-	-	100	61,78	38,22	6,77	5,32	2,73	23,40

> FCI: 0,0353

> TIPI DI RICIRCOLO	PROB	PROB%
riuso/rivendita/redistribuzione diretti	0	0
pulizia e reimballaggio (clean and repackage)	0	0
riparazione (repair)	0,09326	30,01
revisione (refurbishing)	0	0
rilavorazione (remanufacturing)	0,03312	10,66
recupero parti (parts retrieval)	0	0
riciclaggio (recycling)	0	0
incenerimento/smaltimento	0	0
conto-lavoro	0,18437	59,33

Tabelle 3.4 Valori dei principali indici e probabilità di ogni tipologia di ricircolo, calcolati per il network istanza test 2

Network	FCI	TST	AMI	C	A	ϕ	ϕ I	ϕ E	ϕ D	R
Istanza test 1	0,0471	19331	3,1473	97822	60840	36982	6227	5017	2015	23722
Istanza test 2	0,0353	18455	3,0781	91942	56806	35136	6227	4888	2510	21512
Differenza	-0,0118	-876	-0,0692	-5880	-4034	-1846	0	-130	495	-2211
Differenza %	-25,05%	-4,53%	-2,20%	-6,01%	-6,63%	-4,99%	0%	-2,58%	24,55%	-9,32%

Network	% A	% ϕ	% ϕ I	% ϕ E	% ϕ D	% R
Istanza test 1	62,19	37,81	6,37	5,13	2,06	24,25
Istanza test 2	61,78	38,22	6,77	5,32	2,73	23,40
Differenza %	-0,41	0,41	0,41	0,19	0,67	-0,85

Tabelle 3.5 Scostamenti degli indici passando dall'istanza test 1 all'istanza test 2

3.4.Istanze test 3 e 4: network con flussi di materiale riciclabile

La gestione dei Rifiuti da Apparecchiature Elettriche ed Elettroniche (identificati con l'acronimo RAEE[16]) è regolamentata in Italia dal Decreto Legislativo n.151 del 2005, strettamente legato con la Normativa comunitaria RoHS 2002/95/CE per la riduzione di sostanze pericolose nelle medesime apparecchiature.

Nodo centrale del Decreto 151/05 è l'estensione del principio di responsabilità dei produttori anche alle apparecchiature elettriche ed elettroniche con l'obbligo, per le aziende produttrici, di organizzare e gestire un sistema per il riciclo dei prodotti

[16] Noti anche con l'acronimo WEEE dall'inglese *"Waste of electric and electronic equipment"*.

immessi nel mercato e giunti a fine vita: si tratta essenzialmente di una serie di responsabilità logistiche, operative e di controllo lungo tutta la filiera dei RAEE (http://www.ecodom.it).

Oltre al coinvolgimento diretto dei produttori, la normativa attribuisce compiti specifici anche ad altri soggetti coinvolti nella filiera, tra cui i distributori, obbligati a ritirare gratuitamente il RAEE consegnato dai consumatori all'atto dell'acquisto di un'apparecchiatura della medesima tipologia.

Dalla ricerca del 2010 "Gli Italiani nei confronti del recupero e riciclaggio degli elettrodomestici" condotta da Ipsos per Ecodom[17] emerge che solo il 14% delle apparecchiature elettroniche ed informatiche viene ritirato dai negozianti. Del restante 86%, il 67% rimane inutilizzato mentre il 9% viene trattato in modo scorretto (http://www.pubblicaamministrazione.net).

Il Decreto 151/05 stabilisce inoltre gli obiettivi di recupero e di riciclo da raggiungere nella fase di trattamento dei RAEE, a seconda della categoria di prodotto: per quanto riguarda le apparecchiature informatiche e per le telecomunicazioni e l'elettronica di consumo viene imposta una percentuale di recupero pari ad almeno 75% in peso medio per apparecchio e una percentuale di reimpiego e di riciclaggio di componenti, di materiali e di sostanze pari almeno al 65% sempre in peso medio per apparecchio (http://ecolight.nemo.it).

Utilizzando i dati statistici appena presentati e gli obiettivi di recupero imposti dalla normativa vigente che regola la gestione dei RAEE, si costruiscono le istanze test 3 e 4 a partire rispettivamente dalle istanze test 1 e 2. Quest'ultime sono rimodellate in modo da contemplare un sistema di riciclaggio alimentato dal flusso di ritorno dei rifiuti elettronici (PC e apparecchi per la telecomunicazione) consegnati dai consumatori ai *retailer*, i quali sono costretti a ritirarli per ottemperare al cosiddetto "ritiro uno contro uno" dei RAEE previsto dal Decreto 151/05.

Una volta costruite le rispettivi matrici estese T^* per i due casi (tabelle 3.6 e 3.7), si procede con l'analisi del network supportata da software.

[17] *Ipsos* è una società che si occupa di ricerche di mercato, mentre *Ecodom* è un consorzio italiano per il recupero e il riciclaggio di elettrodomestici.

Comparti	Imp.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	Exp.	Diss.
Imp.		300	667	222	114	80			668	143								111	31	98	36	16						
1							285																					15
2								648																				19
3								222																				
4								109																				5
5								72																				8
6											275																	10
7											793			240														18
8												662	157															
9														354														9
10														1056														12
11														645														17
12									151																			6
13															1723	572												
14																	1723											
15																	572											
16																		1050	282	924	337	149						
17																							254			111	796	
18																							62			31	220	
19																							221			98	703	
20																							77			36	260	
21																							32			16	117	
22																	115							273	136			122
23																	245											28
24																	87											49
25										220																		72
Exp.																												
Diss.																												

Tabella 3.6 Matrice estesa dei trasferimenti T^* dell'istanza test 3

Comparti	Imp.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	Exp.	Diss.
Imp.		300	667	222	114	80			668	143								111	31	98	36	16					
1							285																				15
2								648																			19
3								222																			
4								109																			5
5								72																			8
6											275																10
7											793			240													18
8												662	157														
9														354													9
10														1056													12
11														645													17
12									151																		6
13															1723	572											
14																	1723										
15																	572										
16																		1003	272	888	324	140					
17																							108	57	111	796	42
18																							28	11	31	220	13
19																							96	46	98	703	43
20																							30	17	36	260	17
21																							11	5	16	117	7
22																	245										28
23																	87										49
24										220																	72
Exp.																											
Diss.																											

Tabella 3.7 Matrice estesa dei trasferimenti T^* dell'istanza test 4

Si riporta in figura 3.13 il grafo dell'istanza test 3. Come si nota, ogni *retailer* invia i RAEE raccolti (misurati come sempre in tonnellate/anno) al centro di riciclaggio per il loro trattamento. I materiali recuperati vengono quindi inviati ad un fornitore (nodo 9) che di conseguenza si vede diminuire il flusso in entrata della stessa quantità (tonnellate/anno di materiale). Il resto del modello è del tutto uguale al network efficiente dell'istanza test 1.

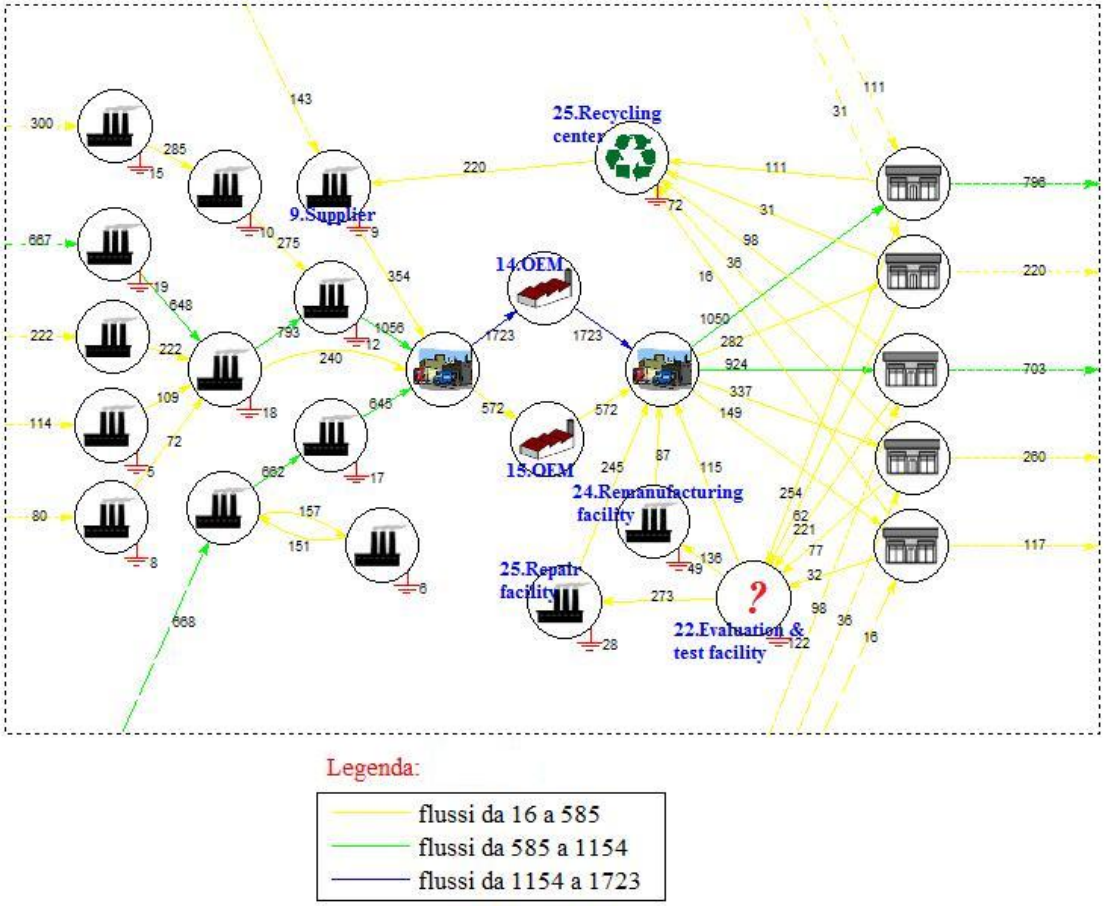

Figura 3.13 Grafo del network istanza test 3

Dall'analisi dei ricircoli, il software ne individua 26: 16 sono i medesimi ricircoli presenti nell'istanza test 1, mentre gli altri 10 sono dovuti al riciclaggio di materiale. Non deve ingannare il fatto che siano solo cinque i flussi che dai punti vendita arrivano al centro di riciclaggio: essendo i ricircoli dei percorsi chiusi, se ne individuano cinque passanti attraverso un produttore (nodo 14), ed altri cinque passanti attraverso l'altro produttore (nodo 15), per un totale di dieci ricircoli dovuti al riciclaggio.

È proprio all'aumentare del numero di nodi e della complessità dei percorsi coinvolti dai ricircoli che emergono i vantaggi nel disporre di un software in grado di svolgere l'intera analisi del network.

Nelle tabelle 3.8 si riportano i risultati ottenuti dalla network analysis dell'istanza test 3. Nelle tabelle 3.9 si leggono invece gli scostamenti percentuali passando dall'istanza test 1 (che non contempla il riciclaggio) all'istanza test 3. L'indice di Finn è aumentato notevolmente (circa del 60%), non tanto per l'entità dei flussi ciclici dovuti al riciclaggio, ma essenzialmente per il numero di attori vi partecipano.

> INDICI:	TST	AMI	C	A	ϕ	ϕ I	ϕ E	ϕ D	R
VALORE:	19987	3,1046	104419	62052	42367	8576	5323	2430	26037
PERCENTUALE:	-	-	100	59,43	40,57	8,21	5,10	2,33	24,94

> FCI: 0,0752

> TIPI DI RICIRCOLO	PROB	PROB%
riuso/rivendita/redistribuzione diretti	0	0
pulizia e reimballaggio (clean and repackage)	0,03791	9,42
riparazione (repair)	0,08075	20,07
revisione (refurbishing)	0	0
rilavorazione (remanufacturing)	0,02868	7,13
recupero parti (parts retrieval)	0	0
riciclaggio (recycling)	0,07071	17,57
incenerimento/smaltimento	0	0
conto-lavoro	0,18437	45,82

Tabelle 3.8 Valori dei principali indici e probabilità di ogni tipologia di ricircolo, calcolati per il network istanza test 3

Network	FCI	TST	AMI	C	A	ϕ	ϕ I	ϕ E	ϕ D	R
Istanza test 1	0,0471	19331	3,1473	97822	60840	36982	6227	5017	2015	23722
Istanza test 3	0,0752	19987	3,1046	104419	62052	42367	8576	5323	2430	26037
Differenza	0,0281	656	-0,0426	6597	1212	5385	2350	306	415	2315
Differenza %	59,66%	3,39%	-1,35%	6,74%	1,99%	14,56%	38%	6,10%	20,57%	9,76%

Network	% A	% ϕ	% ϕ I	% ϕ E	% ϕ D	% R
Istanza test 1	62,19	37,81	6,37	5,13	2,06	24,25
Istanza test 3	59,43	40,57	8,21	5,10	2,33	24,94
Differenza %	-2,77	2,77	1,85	-0,03	0,27	0,68

Tabelle 3.9 Scostamenti degli indici passando dall'istanza test 1 all'istanza test 3

I ricircoli di materiale riciclabile rappresentano la parte di indice di Finn indicativa del grado di sostenibilità della rete industriale. Anche per questo è importante discriminare tra loro le varie tipologie di ricircolo: possiamo così affermare che il 7,52% del materiale (tonnellate/anno) ricircola all'interno dell'istanza test 3, e che il 17,57% di esso, partecipando al ciclo del riciclaggio, autoalimenta in maniera sostenibile l'intera catena di fornitura.

Osservando gli indici entropici si deduce che i flussi diventano più ridondanti (aumenta R) con ripercussioni sul livello di organizzazione del network (diminuisce l'AMI).

Da notare che, nonostante la quantità di materiale e di beni (tonnellate/anno) in entrata nel sistema non sia variata passando all'istanza 3, *l'overhead in import* è cresciuto del 38%: la causa risiede nell'aumento del numero dei flussi provenienti dall'esterno della rete. In altre parole, i RAEE che entrano nel network attraverso i cinque canali situati in corrispondenza dei punti vendita, provocano un aumento di complessità della supply

chain. Tale andamento si riflette solitamente in maggiori costi, giustificati dalle attività supplementari a cui deve sopperire ogni *retailer* per una corretta gestione dei RAEE (che comprende il loro ritiro, lo stoccaggio, etc.).

Infine, confrontando i network attraverso gli indici percentuali pesati sulla capacità del sistema, si vede che passando dall'istanza test 1 alla 2 (medesima configurazione senza il riciclaggio di materiale) diminuisce il rapporto A/C del 2,77%, in quanto il *development capacity* è aumentato in percentuale di più di quanto non abbia fatto l'*ascendancy.*

In figura 3.14 si riporta il grafo del network istanza test 4. Anche in questo caso si vedono i flussi di ritorno dei RAEE (misurati in tonnellate/anno) e i 10 ricircoli dovuti al riciclaggio. Per il resto non cambia nulla dal network reattivo dell'istanza test 2.

Nelle tabelle 3.10 si riportano i risultati dell'analisi del network, mentre nelle tabelle 3.11 si leggono gli scostamenti percentuali passando dall'istanza test 2 (che non prevede il riciclaggio) all'istanza test 4. Da notare che l'indice di Finn è salito a 6,55% e che il 20% del materiale (tonnellate/anno) che ricircola nel sistema rappresenta la quota parte sostenibile dell'intero network.

Per gli indici entropici valgono le stesse considerazioni fatte nel precedente confronto.

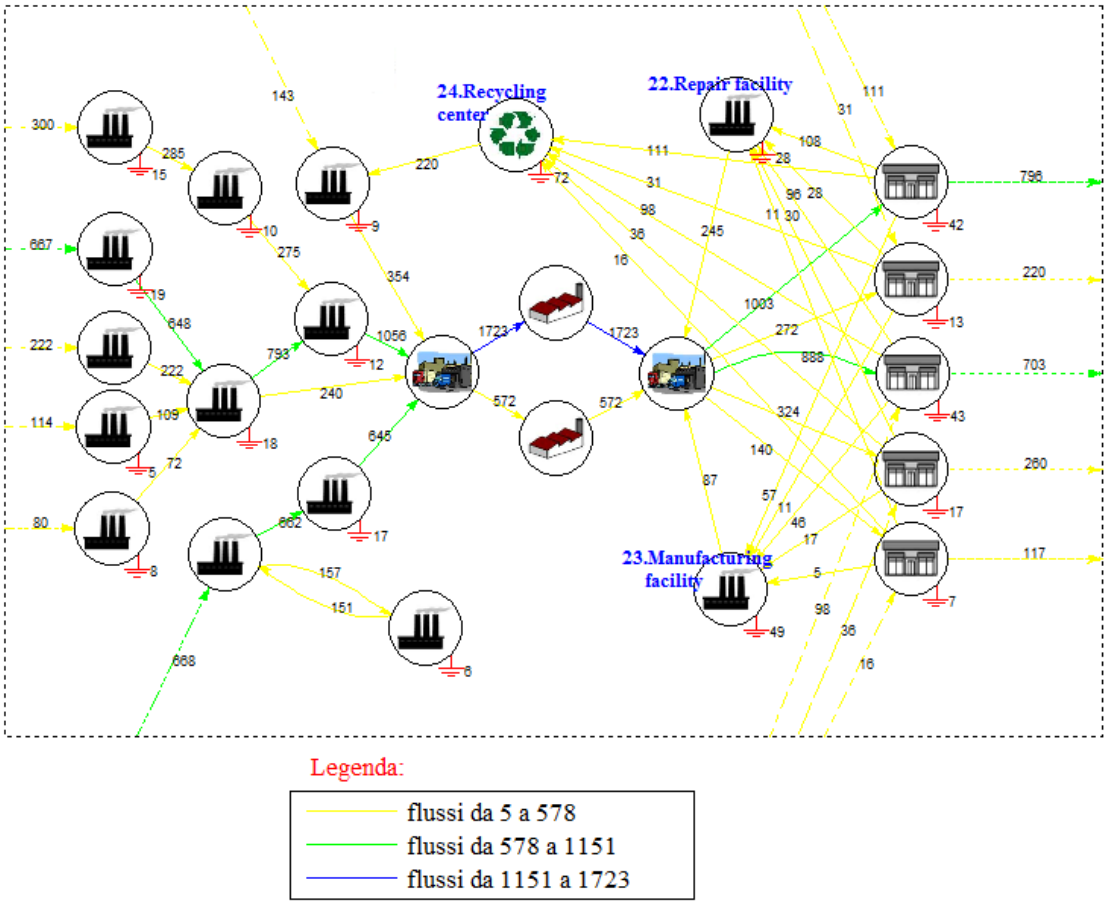

Figura 3.14 Grafo del network istanza test 4

> INDICI:	TST	AMI	C	A	ϕ	ϕ I	ϕ E	ϕ D	R
VALORE:	19111	3,0354	98496	58009	40487	8560	5207	2943	23777
PERCENTUALE:	-	-	100	58,89	41,11	8,69	5,29	2,99	24,14

> FCI: 0,0655

> TIPI DI RICIRCOLO	PROB	PROB%
riuso/rivendita/redistribuzione diretti	0	0
pulizia e reimballaggio (clean and repackage)	0	0
riparazione (repair)	0,08394	22,59
revisione (refurbishing)	0	0
rilavorazione (remanufacturing)	0,02981	8,02
recupero parti (parts retrieval)	0	0
riciclaggio (recycling)	0,07350	19,78
incenerimento/smaltimento	0	0
conto-lavoro	0,18437	49,61

Tabelle 3.10 Valori dei principali indici e probabilità di ogni tipologia di ricircolo, calcolati per il network istanza test 4

Network	FCI	TST	AMI	C	A	ϕ	ϕ I	ϕ E	ϕ D	R
Istanza test 2	0,0353	18455	3,0781	91942	56806	35136	6227	4888	2510	21512
Istanza test 4	0,0655	19111	3,0354	98496	58009	40487	8560	5207	2943	23777
Differenza	0,0302	656	-0,0427	6554	1204	5351	2333	319	433	2266
Differenza %	85,55%	3,55%	-1,39%	7,13%	2,12%	15,23%	37%	6,52%	17,26%	10,53%

Network	% A	% ϕ	% ϕ I	% ϕ E	% ϕ D	% R
Istanza test 2	61,78	38,22	6,77	5,32	2,73	23,40
Istanza test 4	58,89	41,11	8,69	5,29	2,99	24,14
Differenza %	-2,89	2,89	1,92	-0,03	0,26	0,74

Tabelle 3.11 Scostamenti degli indici passando dall'istanza test 2 all'istanza test 4

Nelle tabelle 3.12 sono calcolati gli scostamenti passando dall'istanza test 3 alla 4: il comportamento degli indici è lo stesso riscontrato nel passaggio dall'istanza test 1 alla 2, cioè nel passaggio dal network efficiente a quello reattivo. Tale comportamento è corretto, essendo le prime due istanze speculari rispettivamente alle istanze test 1 e 2, con l'aggiunta dei medesimi flussi di materiale riciclabile.

I due network risultano avere proporzioni di A/C entrambe circa uguali a 59%, in quanto, passando dall'istanza test 1 alla 2, l'*ascendancy* e il *development capacity* diminuiscono di una percentuale molto simile.

Network	FCI	TST	AMI	C	A	ϕ	ϕ I	ϕ E	ϕ D	R
Istanza test 3	0,0752	19987	3,1046	104419	62052	42367	8576	5323	2430	26037
Istanza test 4	0,0655	19111	3,0354	98496	58009	40487	8560	5207	2943	23777
Differenza	-0,0097	-876	-0,0693	-5923	-4043	-1880	-16	-117	513	-2260
Differenza %	-12,90%	-4,38%	-2,23%	-5,67%	-6,52%	-4,44%	0%	-2,20%	21,13%	-8,68%

Network	% A	% ϕ	% ϕ I	% ϕ E	% ϕ D	% R
Istanza test 3	59,43	40,57	8,21	5,10	2,33	24,94
Istanza test 4	58,89	41,11	8,69	5,29	2,99	24,14
Differenza %	-0,53	0,53	0,48	0,19	0,66	-0,80

Tabelle 3.12 Scostamenti degli indici passando dall'istanza test 3 all'istanza test 4

Infine, nelle tabelle 3.13 si riassumono i risultati ottenuti dalle varie supply network analysis: si riportano gli indici entropici, gli indici entropici in percentuale, l'indice di Finn e le probabilità di ogni tipologia di ricircolo, per le quattro istanze test analizzate in questo capitolo di tesi.

Network	FCI	TST	AMI	C	A	ϕ	ϕ I	ϕ E	ϕ D	R
Istanza test 1	0,0471	19331	3,1473	97822	60840	36982	6227	5017	2015	23722
Istanza test 2	0,0353	18455	3,0781	91942	56806	35136	6227	4888	2510	21512
Istanza test 3	0,0752	19987	3,1046	104419	62052	42367	8576	5323	2430	26037
Istanza test 4	0,0655	19111	3,0354	98496	58009	40487	8560	5207	2943	23777

Network	% A	% ϕ	% ϕ I	% ϕ E	% ϕ D	% R
Istanza test 1	62,19	37,81	6,37	5,13	2,06	24,25
Istanza test 2	61,78	38,22	6,77	5,32	2,73	23,40
Istanza test 3	59,43	40,57	8,21	5,10	2,33	24,94
Istanza test 4	58,89	41,11	8,69	5,29	2,99	24,14

	istanza test 1		istanza test 2		istanza test 3		istanza test 4	
> TIPI DI RICIRCOLO	PROB	PROB%	PROB	PROB%	PROB	PROB%	PROB	PROB%
riuso/rivendita/redistribuzione diretti	0	0	0	0	0	0	0	0
pulizia e reimballaggio (clean and repackage)	0,04194	12,07	0	0	0,03791	9,42	0	0
riparazione (repair)	0,08935	25,72	0,09326	30,01	0,08075	20,07	0,08394	22,59
revisione (refurbishing)	0	0	0	0	0	0	0	0
rilavorazione (remanufacturing)	0,03173	9,13	0,03312	10,66	0,02868	7,13	0,02981	8,02
recupero parti (parts retrieval)	0	0	0	0	0	0	0	0
riciclaggio (recycling)	0	0	0	0	0,07071	17,57	0,07350	19,78
incenerimento/smaltimento	0	0	0	0	0	0	0	0
conto-lavoro	0,18437	53,07	0,18437	59,33	0,18437	45,82	0,18437	49,61

Tabelle 3.13 Tabelle riassuntive con gli indici entropici, gli indici entropici in percentuale, l'indice di Finn e le probabilità per tipologia di ricircolo, riportati per tutte le istanze test oggetto di studio

Conclusioni

La network analysis si rivela una metodologia efficace anche per lo studio delle reti di fornitura industriali. Essendo applicata in un contesto diverso da quello ecologico di origine, necessita di essere ancora perfezionata ed ampliata. Tuttavia già allo stato attuale, con gli strumenti presentati in questo lavoro, si riesce a fotografare la struttura dei flussi di un supply network da un'angolazione che permette di cogliere aspetti legati alla complessità, al grado di organizzazione, al fenomeno dei ricircoli e alla sostenibilità, secondo un punto di vista olistico.

Uno dei principali punti di forza della metodologia, e in particolare dell'analisi dei ricircoli, è la semplicità computazionale richiesta: per trovare l'indice di Finn, che fornisce una stima sull'entità dei flussi logistici di ricircolo, è sufficiente calcolare una matrice inversa (la matrice di Leontief). La supply network analysis è di conseguenza una pratica piuttosto veloce, a maggior ragione se si dispone di un software in grado di implementare i principali step della metodologia, dalla mappatura dell'intera rete di fornitura, al calcolo degli indici entropici e di Finn, fino all'individuazione e all'analisi dei ricircoli.

Un'annotazione importante è che permette di condurre un'analisi quantitativa: si ricavano infatti degli indicatori capaci di condensare alcune caratteristiche globali (riguardanti la complessità, ma non solo) di un supply network.

La possibilità di un confronto immediato ed oggettivo degli indici calcolati ha un doppio vantaggio pratico: innanzitutto aiuta nei problemi di ottimizzazione dei flussi logistici che attraversano un network, senza dover intervenire sulla sua struttura (cioè sul numero di attori, sui collegamenti, etc.); secondo, ma più importante aspetto, agevola la progettazione (o la riprogettazione) del network attraverso la comparazione di scenari alternativi.

La configurazione dei flussi ottimale a cui si perviene non è necessariamente quella caratterizzata da una minore complessità (o da un grado maggiore di organizzazione): la valutazione si basa anche su altri aspetti, in particolare sui risultati dell'analisi dei flussi di ricircolo. Infatti, da quanto appurato nello studio delle istanze test proposte nel terzo capitolo della tesi, la presenza di flussi logistici di ricircolo porta sempre ad un aumento di complessità della rete (che si rispecchia in un incremento di *redundancy*): questo non significa che devono essere assolutamente evitati tutti i tipi di ricircolo. Primo perché, eliminando tutti i percorsi ridondanti, si rischia di ottenere supply chain eccessivamente organizzate e di conseguenza troppo fragili a fronte di imprevisti o cambiamenti improvvisi che provocano l'interruzione di qualche flusso. Secondo

perché si adotta un approccio miope riguardo altre questioni di notevole rilevanza, come la sostenibilità di una rete.

Diventa quindi di fondamentale importanza discriminare le varie tipologie di ricircoli riscontrabili all'interno di un supply network (vedi capitolo uno della tesi): se, ad esempio, ce ne sono alcuni dovuti al riciclaggio di materiale, allora la ridondanza intrinseca nei ricircoli stessi può essere giustificata dal fatto che essi rappresentano la parte sostenibile del network, a maggior ragione se tali flussi autoalimentano la rete portando ad una diminuzione di *overhead in import*.

Ovviamente ci potrebbero essere altri motivazioni legate ad esempio ai costi: ricircoli dovuti al riciclaggio garantiscono un ritorno economico associato al recupero e al riutilizzo del materiale all'interno della rete.

Va chiarito che la supply network analysis non è uno studio sulla fattibilità economica, anche se di fatto la complessità si ripercuote sui costi di gestione delle relazioni tra gli attori di una catena di fornitura.

Inoltre, non è illogico pensare di sfruttare la network analysis per delle valutazioni di massima sui costi di gestione della rete: in particolare, se si conoscessero i costi complessivi della logistica dell'intera rete, si potrebbe stimare la parte imputabile alla logistica di ricircolo, moltiplicando i costi totali per l'indice di Finn. Si tratta comunque di un calcolo la cui fattibilità necessita di essere dimostrata attraverso l'impiego di casi studio reali.

In questo lavoro di tesi sono state analizzate delle istanze test che, pur trascinandosi dietro semplificazioni ed approssimazioni tipiche dei modelli ipotetici, sono servite a dimostrare la fattibilità della metodologia: l'obiettivo principale era vedere se le misure ottenute dall'analisi rispecchiavano correttamente le configurazioni dei flussi.

Ovviamente si tratta di risultati preliminari che necessitano di essere verificati su di un numero elevato di configurazioni/reti con dati reali, in modo da osservare il comportamento degli indici e determinare la loro sensibilità di risposta.

Altro aspetto rilevante della supply network analysis è che necessita di un solo input, la matrice estesa dei trasferimenti T^*, che riporta i flussi di scambio tra i vari attori e tra il sistema e l'esterno. D'altronde ciò rappresenta anche il punto debole della metodologia: non sempre tale matrice è disponibile e la sua costruzione può rivelarsi difficoltosa, soprattutto quando si analizzano realtà molto complesse, con un numero elevato di attori e di relazioni.

Inoltre un requisito essenziale per procedere con l'analisi è che la matrice sia in stato di equilibrio (*steady state*). Significa che il flusso di materiale in entrata in ogni nodo deve corrispondere al flusso in uscita dallo stesso, al netto delle perdite, in modo che non ci sia variazione della massa totale. Tuttavia, quando si procede con la raccolta dei dati

per la costruzione di un network, è improbabile che i flussi risultino perfettamente bilanciati.

Un prossimo studio potrebbe incentrarsi proprio sulla risoluzione di questo problema. In ambito ecologico sono già stati sviluppati degli algoritmi per il bilanciamento delle masse (riportati nel lavoro di Allesina (2004)): si potrebbe partire da quelli già disponibili e capire quale di essi meglio si adatta alla realtà industriale.

Altri sviluppi futuri potrebbero continuare con l'estensione e l'adattamento al contesto industriale degli strumenti di network analysis usati per lo studio degli ecosistemi, fino ad implementare tutti gli step dell'analisi. Solo così si può arrivare ad una fotografia completa ed esaustiva sulla struttura dei flussi logistici di un network, che tiene conto di tutte le caratteristiche della rete di relazioni tra i diversi attori della supply chain.

Un'ulteriore considerazione riguarda il fatto che l'analisi svolta è statica, avendo di fatto studiato i flussi che avvengono in un certo periodo di tempo (ad esempio tonnellate/anno). L'ideale sarebbe condurre un'analisi dinamica considerando intervalli limitati di tempo ΔT, per passare da un'istantanea disponibile su lunghi periodi, ad un monitoraggio in tempo reale della supply chain.

In questo lavoro di tesi, il cui focus era lo studio dei flussi logistici di ricircolo, sono stati considerati solo i flussi fisici riguardanti beni, prodotti, materiali, UDC, etc. Tuttavia, alcuni strumenti della network analysis si prestano anche per lo studio di flussi finanziari o di informazioni.

Infine, l'analisi è stata condotta a livello di macro supply network (considerando cioè le relazioni tra organizzazioni), ma potrebbe verosimilmente essere estesa anche a livello micro (riferendosi alle relazioni all'interno delle organizzazioni): i singoli processi che avvengono all'interno di un'azienda possono infatti essere paragonati agli attori di un network, essendo anch'essi caratterizzati da flussi in entrata e in uscita, flussi di scambio con clienti e fornitori interni e dissipazioni.

Appendice

MATLAB è un ambiente per il calcolo numerico e l'analisi statistica che comprende anche l'omonimo linguaggio di programmazione.
MATLAB consente di manipolare matrici, visualizzare funzioni e dati, implementare algoritmi, creare interfacce utente, e interfacciarsi con altri programmi.
In quest'appendice vengono riportate le funzioni del programma sviluppato in MATLAB, che implementa la supply network analysis: dalla mappatura della rete di fornitura al calcolo degli indici entropici, fino all'analisi dei ricircoli.

```
function [ aCicli, pmaxnodo ] = arrayCicli ( T )
%questa funzione restituisce l'array di celle 'aCicli': ogni cella contiene
%un ciclo trovato all'interno della matrice degli scambi (matrice T);
%i cicli sono salvati sottoforma di array numerici in cui i numeri
%rappresentano i nodi che si susseguono formando un ciclo;
%la funzione 'arrayCicli' restituisce anche l'array di celle 'pmaxnodo',
%nelle cui celle è salvato il percorso più lungo a partire da ogni nodo.
global strCicli;
global percorsomax;

strCicli = '';
%trovo il numero di nodi N:
N = length(T);
%creo la matrice di Incidenza (matrice avente stesse dimensioni di T, i cui
%valori sono =1 dove c'è un flusso tra i nodi; =0 dove non c'è flusso)
%MI = logical (T);
MI = ones(size(T));
MI(T==0) = 0;

%NB: potrei semplificare la matrice MI considerando solo i nodi che
%partecipano ai ricircoli (per conoscerli basta osservare la matrice di
%Leontief) -> diminuirebbe considerevolmente la complessità dell'algoritmo;
%tuttavia per conoscere il percorso più lungo di ogni nodo (al fine di
%costruire il grafico) devo passare in rassegna tutti i nodi

for ind = 1:N
    percorsomax = strcat(int2str(ind), ',');
    %richiamo la funzione 'trovaCicli' che restituisce la var 'strCicli'
    %aggiornata e la var stringa 'percorsomax' con il perrcorso più lungo
    %del nodo 'ind' dato in input alla funzione
    [strCicli, percorsomax] = trovaCicli(MI, ind);
    %salvo il percorso max di ogni nodo (che ora è memorizzato nella var
    %'percorsomax' sottoforma di stringa) in una cella della var
    %'pmaxnodo' sottoforma di array numerico:
    pmaxnodo{1, ind} = ( sscanf(percorsomax, ['%d' ',']) );
    %pongo a zero tutti i valori della riga del nodo appena visitato della
    %matrice MI, così da evitare di incappare in loop a causa di cicli
    %MI(ind, :) = 0;
    MI(ind, :) = NaN;
end;

%'strCicli' è una stringa che contiene tutti cicli di seguito separati da ';'
%creo il vettore 'pos' contenente gli indici delle posizioni dei ';'
%all'interno di 'strCicli'
pos = findstr(strCicli, ';');
%'ncelle' è un numero intero pari al numero di cicli presenti in 'cicli'
%che, a sua volta, è uguale alla dimensione di 'pos':
ncelle = length(pos);
%dalla stringa 'strCicli' estrapolo tutti i cicli (sempre sottoforma di
%stringa) in cui i numeri (nodi) sono separati da ',' (es: 2,4,3,2);
%dalle stringhe con i cicli, memorizzate nel vettore cella 'V', estrapolo i
%numeri e li memorizzo nell'array di celle 'aCicli'
V = cell(1, ncelle);
aCicli = cell(1, ncelle);
cont = 0;
```

```
for ind = pos
    cont = cont + 1;
    if cont == 1
        V(1, 1) = {strCicli(1 : ind-1)};
    else
        V(1, cont) = {strCicli(prec : ind-1)};
    end;
    %V{1, cont} = str2num(V{1, cont});  oppure:
    aCicli{1, cont} = (sscanf (V{1, cont}, ['%d' ',']));
    prec = ind + 1;
end;

end
```

```
function [ h, yy, zz ] = arrow ( varargin )

%questa funzione è scaricabile dal sito:
http://www.mathworks.com/matlabcentral/fileexchange/278
```

```
function [ colore, spessore ] = assegnaColore ( freccia )
%questa funzione assegna un colore e uno spessore alla freccia (flusso) in
%input, in base al suo valore
global valMax;
global valMin;
global range;

if freccia < valMin+range
    colore = 'y';
    spessore = 0.8;
elseif freccia < valMax-range
    colore = 'g';
    spessore = 1;
else
    colore = 'b';
    spessore = 1.21;
end;

end
```

```
function [ nodiLiv, nodipercorsomax ] = assegnaLiv ( pmaxnodo, N )
%la funzione 'assegnaLiv' restituisce l'array di celle 'nodiLiv' che
%contiene un numero di celle pari al numero di livelli in cui è suddiviso
%il grafo; in ogni cella salvo i nodi che appartengono rispettivamente al
%livello 1, 2, ..., nLiv

%trovo il percorso 'pmaxnodo' + lungo
pmax = pmaxnodo{1, 1};
for cont = 1: N
    if length(pmaxnodo{1, cont}) > length(pmax)
        pmax = pmaxnodo{1, cont};
    end
end
nodipercorsomax = length(pmax);
%calcolo l'array numerico 'Liv' contenente N valori (pari al numero di
%nodi) in cui ad ogni posizione 'i' è contenuto il livello del grafico in
%cui è posto il nodo 'i'
%es: Liv(3) = 7 -> il nodo 3 si trova in posizione 7 (al 7° liv del grafo);

%ripercorro il percorso max di ogni nodo 'pmaxnodo' al contrario e se trovo
%che un nodo nel percorso considerato ha una posizione (livello) inferiore
%allora assegno al nodo tale posizione (livello) inferiore;

%assegno a tutti i nodi il livello max:
Liv(1:N) = nodipercorsomax;
for cont = 1:N
    nodipercorso = length(pmaxnodo{1, cont});
    %PROCEDURA CONTRARIA (ALLINEO TUTTI I NODI A DESTRA)
    for ind = nodipercorso:-1:1
        %'diff' tiene conto di quanto devo spostare in avanti i nodi del
        %percoso considerato per far in modo di allinearli partendo da dx
        diff = nodipercorsomax - nodipercorso;
        %nella var 'a' trasferisco il percorso (+ lungo) 'pmaxnodo' che
        %parte dal nodo 'cont'
        a = pmaxnodo{1, cont};
        %nella var 'b' salvo ad ogni ciclo un nodo del percorso 'pmaxnodo'
        %che sto considerando
        b = pmaxnodo{1, cont}(ind, 1);
        %'find(a==b)' restituisce la posizione del nodo considerato;
        %ad es se il percorso è [2 4 7 9], e il nodo considerato è 7, mi
        %restituisce la sua posizione (3)
        if ( (find(a==b) + diff) < Liv(b) )
            Liv(b) = find(a==b) + diff;
        end;
    end;
end;
%creo l'array di celle 'nodiLiv' visitando l'array numerico 'Liv'
nodiLiv = cell(1, nodipercorsomax);
for cont = 1:N
    nodiLiv{1, Liv(cont)} = [nodiLiv{1, Liv(cont)}, cont];
end;
end
```

```
function [ indiciPerCiclo ] = calcolaIndiciPerCiclo( matriceCompleta, aCicli )
%questa funzione restituisce l'array di celle 'indiciPerCiclo' contenente
%l'indice TST e gli indici entropici, calcolati per ogni ciclo

%trovo il numero di cicli 'nCicli':
nCicli = length(aCicli);
%creo l'array di celle 'indiciPerCiclo' avente un numero di righe pari a
%'nCicli':
indiciPerCiclo = cell(nCicli, 1);

%per ogni ciclo 'cont' calcolo gli indici e li salvo in una riga di
%'indiciPerCiclo'
for cont = 1:nCicli
    %ad ogni iterazione estrapolo un ciclo presente in 'aCicli' e lo salvo
    %in 'arrayCiclo':
    arrayCiclo = (aCicli{1, cont})';
    %calcolo il numero di nodi 'nNodiCiclo' che compone il ciclo
    %'arrayCiclo':
    nNodiCiclo = length(arrayCiclo)-1;

    %creo la matrice 'matriceCiclo' formata solo dai nodi presenti nel
    %ciclo considerato, con le info su import, export e dissipation: tali
    %matrice verrà data in input alla funzione 'indiciEntropici' che ne
    %calcola TST e gli indici entropici
    matriceCiclo = zeros(nNodiCiclo+3, nNodiCiclo+3);

    %per costruire la matrice 'matriceCiclo' vado a leggere nella
    %'matriceCompleta' i rispettivi flussi tra i nodi del ciclo considerato
    cont1 = 0;
    for riga = 2:nNodiCiclo+1
        cont1 = cont1+1;
        nodoFrom = arrayCiclo(cont1);
        cont2 = 0;
        for colonna = 2:nNodiCiclo+1
            cont2 = cont2+1;
            nodoTo = arrayCiclo(cont2);
            %leggo dalla 'matriceCompleta' il flusso che va da 'nodoFrom' a
            %'nodoTo' e li salvo nella 'matriceCiclo':
            matriceCompleta(nodoFrom+1, nodoTo+1);
            matriceCiclo(riga, colonna) = matriceCompleta(nodoFrom+1, ...
                mnodoTo+1);
        end;
        %Import:
        indNodo = arrayCiclo(cont1)+1;
        matriceCiclo(1, riga) = matriceCompleta(1, indNodo);
        %Export:
        nNodi = length(matriceCompleta)-3;
        matriceCiclo(riga, nNodiCiclo+2) = matriceCompleta(indNodo, nNodi+2);
        %Dissipation:
        matriceCiclo(riga, nNodiCiclo+3) = matriceCompleta(indNodo, nNodi+3);
    end;
    %richiamo la funzione 'indiciEntropici' che mi restituisce gli indici
    %di interesse calcolati dalla 'matriceCiclo':
    [ TST, AMI, C, A, OI, OE, OD, R, ~, ~, ~, ~, ~ ] = ...
        indiciEntropici(matriceCiclo);
```

```
    %salvo in una riga di 'indiciPerCiclo' gli indici appena calcolati:
    indiciPerCiclo{cont, 1} = [{TST}, {AMI}, {C}, {A}, {OI}, {OE}, {OD}, {R}];
end;

end
```

```
function [ H ] = circle ( centro, raggio, colore )
%dati in input il centro (sottoforma di coordinate x,y) e il raggio, la
%funzione 'circle' disegna il cerchio sul grafico

if nargin == 2
    colore = 'k';
end

rad = 2*raggio;
%per tracciare il cerchio uso la funzione 'rectangle' che vuole in input le
%coordinate dell'angolo in basso a sinistra del quadrato circoscritto:
n_coor = centro-[raggio raggio];
H = rectangle ('position', [n_coor, rad, rad], 'curvature', [1, 1],
'LineWidth', 0.6 , 'EdgeColor', colore);
daspect([1 1 1]);

end
```

```
function [ C ] = coordCerchi ( C, T )
%questa funzione ha lo scopo di collocare i cerchi nel grafico, assegnando
%ad ognuno di essi una coppia di coordinate (x, y) secondo qualche
%criterio; in input passo la matrice C, nelle cui celle della 1° riga
%verranno assegnate le coordinate per ogni vertice.
global aCicli;

%trovo il numero di nodi 'N'
[~,N] = size(C);
%imposto lo spazio tra i livelli nel grafo
spazioLivelli = 4;
%chiamo la funzione 'arrayCicli' che mi restituisce l'array di celle
%'aCicli' contenente i cicli e l'array di celle 'pmaxnodo' contenente il
%percorso massimo di ogni nodo; anche se per fare il grafico uso solo
%'pmaxnodo', dichiaro 'aCicli' var globale così quando mi serve è già
%disponibile senza dover richiamare la funzione 'arrayCicli'
[aCicli, pmaxnodo] = arrayCicli(T);
%la funzione 'assegnaLiv' restituisce l'array di celle 'nodiLiv' con numero
%di celle pari al numero di livelli in cui è suddiviso il grafo; in ogni
%cella son contenuti i nodi che appartengono rispettivamente al livello 1,
%2, ..., nLiv
[nodiLiv, nLiv] = assegnaLiv(pmaxnodo, N);
%trovo l'ordinata max 'ymax', pari al numero di nodi max nel livello con
%più nodi, moltiplicati * 4
ymax = 0;
for cont = 1:nLiv
    if ( length( nodiLiv{1, cont} ) > ymax )
        ymax = length( nodiLiv{1, cont} );
    end;
end;
ymax = ymax * 4;

%assegno le coordinate del centro di ogni nodo, salvandole nella prima riga
%della matrice C
x = 0;
for cont = 1:nLiv
    y = 0;
    incy = ymax / ( length(nodiLiv{1, cont}) + 1 );
    x = x + spazioLivelli;
    lunghezza = length(nodiLiv{1, cont});
    for ind = lunghezza:-1:1
        y = y + incy;
        nodo = nodiLiv{1, cont}(1, ind);
        C{1,nodo}(1,1) = x;
        C{1,nodo}(1,2) = y;
    end;
    %prima cella della riga 7 della matrice C
    C{7,1}(1,1) = nLiv*spazioLivelli + spazioLivelli;
    C{7,1}(1,2) = ymax;
end;

end
```

```
function [ ] = creaGrafico ( C )
%data in input la matrice C, questa funzione crea il grafico finale
global T;
global I;
global E;
global D;
global fig;
global valMax;
global valMin;
global range;
global spazioLivelli;

if nargin == 0
    fig = figure ('Name', 'Grafo', 'Color', [1 1 1]);
    set(fig, 'units', 'normalized', 'position', [0.02 0.07 0.6 0.8]);
    spazioLivelli = 4;
    [C] = creaMatriceC (T, I, E, D);
end;

hold on;
axis off;
%delimito l'area del grafico impostando i val max delle ordinate e ascisse;
xmax = C{7,1}(1,1);
ymax = C{7,1}(1,2);
axis([-1, xmax+1, -1, ymax+1]);
%congelo i limiti degli assi:
axis manual;
axis(axis);
%fisso il raggio dei cerchi che andrò a creare:
raggio=1;
[~, N] = size (C);
%creo i cerchi richiamando la funzione 'circle(coord_centro, raggio)';
%le coordinate dei centri dei cerchi sono memorizzati in sequenza nella
%prima riga (celle) della matrice C;
for cont = 1:N
    circle(C{1,cont}, raggio);
    text(C{1,cont}(1,1), C{1,cont}(1,2), int2str(cont), 'FontName', 'Impact',
'HorizontalAlignment', 'center');
end;

%da ogni cerchio faccio partire le frecce verso gli altri nodi memorizzati
%nella seconda riga (nelle celle dei rispettivi nodi) della matrice C;
%con la funzione 'estremita(centro_cerchio_in, centro_cerchio_fin, raggio)'
%trovo le coordinate di inizio e fine della freccia, in modo che tocchi
%le circonferenze; con la funzione 'arrow' disegno la freccia;

%si vuole colorare le frecce (e assegnarne uno spessore) in base all'entità
%del flusso: la funzione 'assegnaColore' assegna 3 diversi colori a seconda
%del range a cui appartiene il flusso; il range è stabilito suddividendo
%l'intervallo dato da (flusso max - flusso min)/3
%non considero la dissipation nel calcolo del range:
A = [T, I', E];
valMax = max( A(A>0) );
valMin = min( A(A>0) );
range = (valMax-valMin)/3;
```

```
clear A;
for cont1 = 1:N
    %disegno le frecce dei flussi interni
    C1 = C{1,cont1};
    nfrecce = length(C{2,cont1});
    for cont2 = 1 : nfrecce
        C2 = C{1, C{2,cont1}(1,cont2)};
        [A,B] = estremita(C1, C2, raggio);
        %assegno al flusso un colore e uno spessore richiamando la funzione
        %'assegnaColore'
        %flusso: T(cont1, C{2,cont1}(1,cont2))
        [colore, spessore] = assegnaColore( T(cont1, C{2,cont1}(1,cont2)) );
        %se esiste un flusso di ritorno, oppure se i nodi hanno la stessa
        %ascissa (sono nello stesso livello) o se hanno la stessa ordinata,
        %al posto di una freccia disegno un arco:
        if ( T(C{2,cont1}(1,cont2), cont1) ~= 0 ) || ( A(1)==B(1) ) || ...
            ( A(2)==B(2) )
            AB = sqrt( (A(1)-B(1))^2 +(A(2)-B(2))^2 );
            AO = AB;
            AC = AB/2;
            OC = sqrt( (AO^2) - (AC^2) );
            PC = AO-OC;
            Cy = (A(2)+B(2))/2;
            Cx = (A(1)+B(1))/2;
            alfa = atan2( (B(1)-A(1)), (B(2)-A(2)) );
            P(1) = Cx + PC*cos(alfa);
            P(2) = Cy + PC*sin(alfa);
            xy = [];
            xy = [A(1), P(1), B(1); A(2), P(2), B(2)];
            t = 1:3;
            ts = 1: 0.1: 3;
            xys = spline(t,xy,ts);
            %plot the interpolated curve.
            plot(xys(1,:),xys(2,:), 'LineWidth', spessore, 'Color', colore);
            %disegno la punta della freccia:
            R(1) = B(1) + ((P(1)-B(1))/10000);
            R(2) = B(2) + ((P(2)-B(2))/10000);
            arrow(R, B, 8, 'BaseAngle', 60, 'LineWidth', spessore, ...
                'FaceColor', colore, 'EdgeColor', colore);
        else
            arrow(A, B, 8, 'BaseAngle', 60, 'LineWidth', spessore, ...
                'FaceColor', colore, 'EdgeColor', colore);
        end;
    end;

    %disegno le frecce in 'Import'
    if C{4,cont1} ~= 0
        h1 = ymax - C{1,cont1}(1,2);
        h2 = C{1,cont1}(1,2);
        if C{1,cont1}(1,1) <= spazioLivelli*1.8
            yf = C{1,cont1}(1,2);
            xf = 0;
        elseif h1<h2
            yf = ymax;
            xf = C{1,cont1}(1,1) - spazioLivelli*( 0.3 + (h1/(ymax/2)) );
```

```
        else
            yf = 0;
            xf = C{1,cont1}(1,1) - spazioLivelli*( 0.3 + (h2/(ymax/2)) );
        end;
        [~,B] = estremita([xf, yf], C1, raggio);
        %flusso: I(1, cont1)
        [colore, spessore] = assegnaColore( I(1, cont1) );
        arrow([xf, yf], B, 10, 'LineStyle', '-.', 'BaseAngle', 60, ...
            'LineWidth', spessore, 'FaceColor', colore, 'EdgeColor', colore);
    end;

    %disegno le frecce in 'Export'
    if C{5,cont1} ~= 0
        h1 = ymax - C{1,cont1}(1,2);
        h2 = C{1,cont1}(1,2);
        if C{1,cont1}(1,1) >= xmax-spazioLivelli*1.8
            yf = C{1,cont1}(1,2);
            xf = xmax;
        elseif h1 < h2
            yf = ymax;
            xf = C{1,cont1}(1,1) + spazioLivelli*( 0.3 + (h1/(ymax/2)) );
        else
            yf = 0;
            xf = C{1,cont1}(1,1) + spazioLivelli*( 0.3 + (h2/(ymax/2)) );
        end;
        [~,B] = estremita([xf, yf], C1, raggio);
        %flusso: E(cont1, 1)
        [colore, spessore] = assegnaColore( E(cont1, 1) );
        arrow(B, [xf, yf], 10, 'LineStyle', ':', 'BaseAngle', 60, ...
            'LineWidth', spessore, 'FaceColor', colore, 'EdgeColor', colore);
    end;

    %disegno le dissipazioni 'Dissipation'
    if C{6,cont1} ~= 0
        xd = C{1,cont1}(1,1) + raggio* sqrt(2)/2;
        yd = C{1,cont1}(1,2) - raggio* sqrt(2)/2;
        line([xd xd],[yd yd-0.4], 'Color', 'r');
        line([xd-0.35 xd+0.35],[yd-0.4 yd-0.4], 'Color', 'r');
        line([xd-0.25 xd+0.25],[yd-0.5 yd-0.5], 'Color', 'r');
        line([xd-0.1 xd+0.1],[yd-0.6 yd-0.6], 'Color', 'r');
    end;

end;

%legenda:
a = valMin;
b = round((valMin+range)*100) / 100;
c = round((valMax-range)*100) / 100;
d = valMax;
h1 = line('Color', 'y', 'LineWidth', 0.8);
stringa1 = strcat('flussi da ''', num2str(a), ''' a ''', num2str(b), '''');
h2 = line('Color', 'g', 'LineWidth', 1);
stringa2 = strcat('flussi da ''', num2str(b), ''' a ''', num2str(c), '''');
h3 = line('Color', 'b', 'LineWidth', 1.21);
stringa3 = strcat('flussi da ''', num2str(c), ''' a ''', num2str(d), '''');
```

```
legenda = legend([h1, h2, h3], stringa1, stringa2, stringa3);
line('Color', 'w', 'LineWidth', 0.8);
set(legenda, 'FontName', 'Times New Roman', 'FontSize', 10);
set(legenda, 'location', 'NorthEast');
titolo = get(legenda, 'title');
set(titolo, 'String', 'Legenda: ', 'Color', 'r', ...
    'FontName', 'Times New Roman', 'FontSize', 10);
%titolo:
text(2, ymax+2, 'GRAFO:', 'FontName', 'Algerian', 'FontSize', 20, ...
    'Color', 'r');
%disegno un rettangolo tratteggiato che delimita l'area del grafo:
rectangle('Position', [0, 0, xmax, ymax], 'LineStyle', ':');
%fisso il grafo in modo che non sia possibile spostarlo all'interno della
%figura
ax = gca;
h = pan;
setAllowAxesPan(h,ax,false);
hold off;

end
```

```
function [ C ] = creaMatriceC ( T, I, E, D )
%questa funzione crea la matrice C che serve per creare il grafico
global C;

%trovo il numero di nodi N;
N = length(T);
%creo un array di celle in cui ogni nodo i corrisponde alla colonna i-esima
%e vado a memorizzare, all'interno della matrice C:
% - sulla riga 1 le coordinate dei centri per ogni nodo i;
% - sulla riga 2 i nodi a cui è collegato il nodo i-esimo;
% - sulla riga 3 i flussi dal nodo i-esimo;
% - sulla riga 4 i flussi in ingresso;
% - sulla riga 5 i flussi in uscita;
% - sulla riga 6 i flussi dissipati;
% - sulla prima cella della riga 7 salvo la coordinata max di x e di y
C = cell(7,N);

%riga 1 della matrice C
%la funzione 'coordCerchi' decide dove posizionare il cerchio (nodo)
%all'interno del grafico e salva le coordinate del centro di ogni cerchio
%nella matrice C;
[C] = coordCerchi(C, T);
for cont1 = 1:N
    ind = 0;
    for cont2 = 1:N
        if T(cont1,cont2) ~= 0
            ind = ind+1;
            %riga 2 della matrice C
            C{2,cont1}(1,ind) = cont2;
            %riga 3 della matrice C
            C{3,cont1}(1,ind) = T(cont1,cont2);
        end;
    end;
    %riga 4 della matrice C
    C{4,cont1} = I(1,cont1);
    %riga 5 della matrice C
    C{5,cont1} = E(cont1,1);
    %riga 6 della matrice C
    C{6,cont1} = D(cont1,1);
 end;

 end
```

```
function [ T, I, E, D ] = creaMatriciTIED ( percorsoFile )
%questa funzione crea le matrici T I E D dando eventualmente in input il
%percorso con il file excel con la matrice dei flussi
global T;
global I;
global E;
global D;

%se chiamando la funzione passo già il percorso del file allora non chiedo
%all'utente di inserire il percorso con il file, altrimenti sì
if nargin == 1
    percorso = percorsoFile;
else
    percorso = input('inserisci percorso file excel: ');
end;

if exist(percorso, 'file')
    [A, B] = xlsread(percorso, 'Foglio1');
    %~
    [~, nColonneB] = size(B);
    N = nColonneB - 4;
    A(1, :) = [];
    A(:, 1:2) = [];
    [~, nColonneA] = size(A);
    %se la colonna 'Dissipation' è vuota la rimpiazzo con zeri (idem per la
    %colonna 'Export':
    while nColonneA < N+2
        A(:, nColonneA+1) = 0;
        [~, nColonneA] = size(A);
    end;
    %sostituisco le celle 'NaN' della matrice A (presenti nel caso in cui la
    %matrice importata aveva delle celle vuote) con '0':
    A(isnan(A)) = 0;

    %matrice IMPORT:
    [I] = A(1, 1:N);
    %matrice EXPORT:
    [E] = A(2:N+1, N+1);
    %matrice DISSIPATION:
    [D] = A(2:N+1, N+2);
    %matrice dei flussi interni EXCHANGE:
    [T] = A(2:N+1, 1:N);
else
    disp('File non trovato');
end;

end
```

```
function [ A, B ] = estremita ( C1, C2, raggio )
%questa funzione restituisce le estremità A(x,y) e B(x,y) che appartengono
%rispettivamente al cerchio C1 e al cerchio C2: tali estremi saranno
%l'inizio e la fine della freccia (input della funzione 'arrow')

Y = C2(2) - C1(2);
X = C2(1) - C1(1);
ang = atan2(C2(2)-C1(2), C2(1)-C1(1));
xd = raggio * cos(ang);
yd = raggio * sin(ang);
A = [C1(1)+xd, C1(2)+yd];
B = [C2(1)-xd, C2(2)-yd];

end
```

```
function varargout = finestraIndici(varargin)
% FINESTRAINDICI M-file for finestraIndici.fig
% Last Modified by GUIDE v2.5 25-Aug-2011 22:57:34
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @finestraIndici_OpeningFcn, ...
                   'gui_OutputFcn',  @finestraIndici_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before finestraIndici is made visible.
function finestraIndici_OpeningFcn(hObject, eventdata, handles, varargin)
global aCicli;

%posiziono la finestra nello schermo
set(handles.finestraIndici, 'Units', 'Normalized', ...
    'Position', [0.35 0.3 0.3 0.46]);
%se non ci sono cicli disabilito 'cbProbCicli'
if isempty(aCicli)
    set(handles.cbProbCicli, 'Value', 0);
    set(handles.cbProbCicli, 'Enable', 'off');
else
    set(handles.cbProbCicli, 'Enable', 'on');
end;

% Choose default command line output for finestraIndici
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);


% --- Outputs from this function are returned to the command line.
function varargout = finestraIndici_OutputFcn(hObject, eventdata, handles)
% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in cbG.
function cbG_Callback(hObject, eventdata, handles)

% --- Executes on button press in cbTST.
function cbTST_Callback(hObject, eventdata, handles)
```

```
% --- Executes on button press in cbL.
function cbL_Callback(hObject, eventdata, handles)


% --- Executes on button press in cbCI.
function cbCI_Callback(hObject, eventdata, handles)


% --- Executes on button press in cbProbCicli.
function cbProbCicli_Callback(hObject, eventdata, handles)
global infoRicircolo

%controllo se mancano info sui cicli, guardando se 'infoRicircolo' contiene
%celle vuote
if ( sum(cellfun('isempty', infoRicircolo(1, :))) > 0 )
    warndlg('Info sui cicli non complete', 'ATTENZIONE');
end;


% --- Executes on button press in buttonCalcola.
function buttonCalcola_Callback(hObject, eventdata, handles)
global aCicli;
global T;
global I;
global E;
global D;
global infoRicircolo;
global fig;

%calcolo il numero di nodi N
N = length(T);
%ricalcolo la matrice completa:
matriceCompleta = zeros(N+3, N+3);
matriceCompleta(2:N+1, 2:N+3) = [T, E, D];
matriceCompleta(1, 2:N+1) = [I];
%richiamo la funzione 'indiciEntropici' che mi restituisce i seguenti
%finestraindici:
[ TST, AMI, C, A, OI, OE, OD, R, A_C, OI_C, OE_C, OD_C, R_C ] = ...
    indiciEntropici(matriceCompleta);
%richiamo la funzione 'indiciCicli' che mi restituisce i seguenti indici:
[ G, L, CI, prob, perc ] = indiciCicli(T, I, E, D, aCicli);
%trovo il numero di cicli 'nCicli'
nCicli = length(aCicli);
%guardo se nella cartella 'risultati' esiste già un file 'risultati.xls' e,
%se c'è, lo cancello
oldFolder = pwd;
cd ..;
percorsoFileRisultati = pwd;
percorsoFileRisultati = strcat(percorsoFileRisultati, ...
    '\risultati\risultati.xls');
cd(oldFolder);
if exist(percorsoFileRisultati)
    warndlg('Attenzione, il file ''risultati'' sarà sovrascritto', ...
```

```
        'Attenzione');
    delete(percorsoFileRisultati);
end;
rigaExcel = 1;
vett = [1:N];

%scrivo nel file excel 'risultati.xls' la matrice G:
if get(handles.cbG, 'Value') == 1
    label = cell(N+2, N+1);
    label(1, 1) = {'MATRICE G:'};
    label(2, 2:N+1) = num2cell(vett);
    label(3:N+2, 2:N+1) = num2cell(G);
    label(3:N+2, 1) = num2cell(vett');
    cellaExcel = strcat('A', num2str(rigaExcel));
    xlswrite(percorsoFileRisultati, label, 'Foglio1', cellaExcel);
    rigaExcel = rigaExcel+N+4;
    clear label;
end;

%scrivo nel file excel 'risultati.xls' la matrice L:
if get(handles.cbL, 'Value') == 1
    label = cell(N+2, N+1);
    label(1, 1) = {'MATRICE L:'};
    label(2, 2:N+1) = num2cell(vett);
    label(3:N+2, 2:N+1) = num2cell(L);
    label(3:N+2, 1) = num2cell(vett');
    cellaExcel = strcat('A', num2str(rigaExcel));
    xlswrite(percorsoFileRisultati, label, 'Foglio1', cellaExcel);
    rigaExcel = rigaExcel+N+4;
    clear label;
end;

%scrivo nel file excel 'risultati.xls' l'indice TST e gli finestraindici
entropici:
if get(handles.cbTST, 'Value') == 1
    %creo una nuova figura nella quale vado a rappresentare, mediante un
    %diagramma a torte, le percentuali di Ascendancy e di Overhead;
    %in una barra rappresento, invece, la suddivisione dell'Overhead nelle
    %sue 4 componenti (Overhead in Import, Export, Dissipation e Redundancy).
    %creo la nuova figura 'diagr':
    diagr = figure(fig+1);
    set(diagr, 'Color', [1 1 1], 'Name', 'Diagrammi');
    colormap jet;
    %diagramma a torta
    s1 = subplot(1,4,1:2);
    x = [A, C-A];
    explode = [1 0];
    pie(x,explode)
    %diagramma a barre
    s2 = subplot(1,4,3);
    x = [OI, OE, OD, R];
    barra = bar([x; nan(size(x))], 'stack');
    set(barra(1),'FaceColor', 'g');
    set(barra(2),'FaceColor', 'r');
    set(barra(3),'FaceColor', 'y');
```

```
    set(barra(4),'FaceColor', 'm');
    xlim([.5, 1.5]);
    axis off;
    %creo le legende dei diagrammi appena realizzati
    legend1 = legend(s1, 'show', '%Ascendency', '%Overhead');
    titolo1 = get(legend1, 'title');
    set(titolo1, 'string', 'Capacity:', 'FontSize', 12);
    set(legend1, 'Units', 'normalized');
    set(legend1, 'position', [0.8, 0.65, 0.09, 0.2]);
    legend2 = legend(s2, 'show', '%Import', '%Export', '%Dissipation', ...
        '%Redundancy');
    titolo2 = get(legend2, 'title');
    set(titolo2, 'string', 'Overhead in:', 'FontSize', 12);
    set(legend2, 'Units', 'normalized');
    set(legend2, 'position', [0.8, 0.12, 0.09, 0.4]);
    %salvo la figura con i diagrammi nella cartella 'risultati',
    %nominandola 'diagrammiIndiciEntropici.jpg'
    oldFolder = pwd;
    cd ..;
    percorsoFileDiagr = pwd;
    percorsoFileDiagr = strcat(percorsoFileDiagr, ...
    '\risultati\diagrammiIndiciEntropici.jpg');
    cd(oldFolder);
    saveas(diagr,percorsoFileDiagr);
    %scrivo nel file excel 'risultati.xls' gli indici
    label = cell(3, 10);
    label(1, :) = [{'> INDICI:'}, {'TST'}, {'AMI'}, {'C'}, {'A'}, ...
        {'Overhead'}, {'Overhead I'}, {'Overhead E'}, {'Overhead D'}, {'R'}];
    label(2, :) = [{'VALUE:'}, {TST}, {AMI}, {C}, {A}, {C-A}, {OI}, ...
        {OE}, {OD}, {R}];
    label(3, :) = [{'PERCENTUALE:'}, {'-'}, {'-'}, {100}, {A_C}, ...
        {100-A_C}, {OI_C}, {OE_C}, {OD_C}, {R_C}];
    cellaExcel = strcat('A', num2str(rigaExcel));
    xlswrite(percorsoFileRisultati, label, 'Foglio1', cellaExcel);
    rigaExcel = rigaExcel+4;
    clear label;
end;

%scrivo nel file excel 'risultati.xls' l'indice di Finn CI:
if get(handles.cbCI, 'Value') == 1
    label = [{'FCI:'}, {CI}];
    cellaExcel = strcat('A', num2str(rigaExcel));
    xlswrite(percorsoFileRisultati, label, 'Foglio1', cellaExcel);
    rigaExcel = rigaExcel+2;
    clear label;
end;

%scrivo nel file excel 'risultati.xls' la prob di ogni ciclo trovato, il
%tipo e la prob per ogni tipo di ciclo:
if get(handles.cbProbCicli, 'Value') == 1
    %memorizzo nell'array di celle 'string' i cicli trovati, sottoforma di
    %stringhe:
    string = cell(1, nCicli);
    for cont = 1:nCicli
        string{1, cont} = '';
```

```
    narr = length(aCicli{1, cont});
    for cont2 = 1:narr
        string{1, cont} = strcat( string{1, cont}, '->', ...
            num2str(aCicli{1, cont}(cont2)) );
    end;
    string{1, cont}(1:2) = '';
end;
[indiciPerCiclo] = calcolaIndiciPerCiclo( matriceCompleta, aCicli );
label = cell(nCicli+1, 11);
label(1, :) = [{'> CICLI'}, {'PROBABILITA'''}, {'PROBABILITA'' %'}, ...
    {'TST'}, {'AMI'}, {'C'}, {'A'}, {'Overhead I'}, {'Overhead E'}, ...
    {'Overhead D'}, {'R'}];
label(2:nCicli+1, 1:3) = [(string)', num2cell(prob)', num2cell(perc)'];
cont = 0;
for riga = 2:nCicli+1
    cont = cont+1;
    label(riga, 4:11) = indiciPerCiclo{cont, :};
end;
cellaExcel = strcat('A', num2str(rigaExcel));
xlswrite(percorsoFileRisultati, label, 'Foglio1', cellaExcel);
rigaExcel = rigaExcel+nCicli+2;
clear label;
%leggo le categorie dal file 'popupCatRicircolo.txt'
oldFolder = pwd;
cd ..;
percorsoFile = pwd;
percorsoFile = strcat(percorsoFile, '\icone+popup\popupCatRicircolo.txt');
cd(oldFolder);
%apro l'archivio in modalità di lettura:
file = fopen(percorsoFile, 'r');
%salvo nell'array di celle 'stTipi' il contenuto del file:
stTipi = textscan(file, '%s','delimiter', '\n');
%trovo il numero di categorie salvate nel file:
nTipi = length(stTipi{1,1});
%creo l'array di celle 'stringaTipiCiclo', avente dimensione nx3: nella
%1° colonna salvo le categorie, nella 2° la probabilità di ogni categoria
%e nella 3° la percentuale di ogni categoria
stringaTipiCiclo = cell(nTipi, 3);
stringaTipiCiclo(1:nTipi, 1) = stTipi{1,1};
%l'ultima riga della prima colonna la riservo ai cicli dei quali non è
%stato indicato il tipo:
stringaTipiCiclo{nTipi, 1} = ['-'];
%pongo la 2° e la 3° colonna di 'stringaTipiCiclo' = 0:
stringaTipiCiclo(:, 2:3) = {0};
%completo la 2° e la 3° colonna di 'stringaTipiCiclo';
%passo in rassegna tutti i cicli:
for contCiclo = 1:nCicli
    %da 'infoRicircolo' ricavo la categoria del ciclo considerato e la
    %salvo su 'tipoRicircolo':
    tipoRicircolo = infoRicircolo{1, contCiclo};
    %cerco nella 1° colonna di 'stringaTipiCiclo' la riga
    %'posizioneTipo' che corrisponde a 'tipoRicircolo'
    %se non è stata inserita l'info sul tipo di ciclo allora
    %'posizioneTipo' corrisponde all'ultima riga:
    if ~isempty(tipoRicircolo)
```

```
            posizioneTipo = find(strcmp(stringaTipiCiclo(:,1), ...
                tipoRicircolo));
        else
            posizioneTipo = nTipi;
        end;
        %nella riga 'posizioneTipo' della 2° e 3° colonna di
        %'stringaTipiCiclo', sommo a quello già presente, rispettivamente
        %la probabilità 'prob' e la percentuale 'perc' calcolate in
        %precedenza:
        stringaTipiCiclo(posizioneTipo, 2) = ...
            {stringaTipiCiclo{posizioneTipo, 2} + prob(1, contCiclo)};
        stringaTipiCiclo(posizioneTipo, 3) = ...
            {stringaTipiCiclo{posizioneTipo, 3} + perc(1, contCiclo)};
    end;
    %chiudo l'archivio
    fclose(file);
    %se non ci sono cicli mancanti dell'info sulla categoria allora
    %cancello l'ultima riga di 'stringaTipiCiclo'
    if stringaTipiCiclo{nTipi, 2} == 0
        stringaTipiCiclo(nTipi, :) = [];
    end;
    %tabella con i tipi di ciclo e la loro prob:
    label = [{'> TIPI DI CICLO'}, {'PROBABILITA'''}, {'PROBABILITA'' %'}];
    label = [label; stringaTipiCiclo];
    cellaExcel = strcat('A', num2str(rigaExcel));
    xlswrite(percorsoFileRisultati, label, 'Foglio1', cellaExcel);
    rigaExcel = rigaExcel+nTipi+2;
    clear label;

end;
finestraIndici_CloseRequestFcn(finestraIndici);


% --- Executes when user attempts to close finestraIndici.
function finestraIndici_CloseRequestFcn(hObject, eventdata, handles)
% Hint: delete(hObject) closes the figure
delete(hObject);
```

```
function varargout = finestraInfoGrafo(varargin)
% FINESTRAINFOGRAFO M-file for finestraInfoGrafo.fig
% Last Modified by GUIDE v2.5 26-Aug-2011 11:47:53
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @finestraInfoGrafo_OpeningFcn, ...
                   'gui_OutputFcn',  @finestraInfoGrafo_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before finestraInfoGrafo is made visible.
function finestraInfoGrafo_OpeningFcn(hObject, eventdata, handles, varargin)
global T;
global I;
global E;
global D;
global nodoSelPrec;
global vettNodiSteady;

%posiziono la finestra nello schermo
set(handles.finestraInfoGrafo, 'Units', 'Normalized', 'Position', ...
     [0.65 0.28 0.33 0.72]);
%imposto le proprietà di slider 'sliderDimNum' che va da un valore (Font
%Size) minimo di 4 ad un max di 14
slider_step(1:2) = 1/(14-4);
set(handles.sliderDimNum,'sliderstep',slider_step, ...
    'max', 14, 'min', 4, 'Value', 5);
%imposto la visualizzazione degli oggetti nel panel 'panelFlussi':
set(handles.textFlussoVersoNodo, 'String', ' ');
set(handles.textFlussoImport, 'String', ' ');
set(handles.textFlussoExport, 'String', ' ');
set(handles.textFlussoDissipato, 'String', ' ');
set(handles.popupVersoNodo, 'String', ' ');
ObjPanelFlussi = [handles.textVersoNodo, handles.textImport, ...
    handles.textExport, handles.textDissipato, handles.popupVersoNodo, ...
    handles.text2punti, handles.textFlussoVersoNodo, ...
    handles.textFlussoImport, handles.textFlussoExport, ...
    handles.textFlussoDissipato];
set(ObjPanelFlussi, 'Enable', 'off');
%imposto la visualizzazione dei 'panelInfoNodo' e di 'editAltro':
set(handles.panelInfoNodo, 'Visible', 'off');
set(handles.editAltro, 'Visible', 'off');
```

```
%inserisco nel popup 'popupNodoSel' la lista con tutti i nodi, creando la
%stringa 'stringaNodi' con la sequenza dei nodi separati da '|', che il
%popup interpreta come andare a capo:
N = length(T);
stringaNodi = '';
for cont = 1:N
    stringaNodi = strcat(stringaNodi, '|', num2str(cont));
end;
set(handles.popupNodoSel, 'String', stringaNodi);

%la var 'nodoSelPrec' contiene il nodo selezionato in precedenza:
%all'inizio lo imposto = 0;
nodoSelPrec = 0;
%leggo dal file 'popupCatNodo.txt' (che deve trovarsi nella cartella
%'icone+popup' che a sua volta deve trovarsi nella stessa directory della
%cartella 'funzioni') le categorie e le carico nel popup 'popupCatNodo'
oldFolder = pwd;
cd ..;
percorsoFile = pwd;
percorsoFile = strcat(percorsoFile, '\icone+popup\popupCatNodo.txt');
cd(oldFolder);
%apro l'archivio in modalità di lettura
file = fopen(percorsoFile, 'r');
%salvo nell'array di celle 'stringaCat' il contenuto del file
stringaCat = textscan(file, '%s','delimiter', '\n');
stringaCat = stringaCat{1};
%carico nel popup 'popupCatNodo' le categorie contenute in 'stringaCat'
set(handles.popupCatNodo, 'String', stringaCat);
%chiudo l'archivio
fclose(file);
%trovo i nodi che sono in 'steady state' e salvo il loro stato nel vettore
%'vettNodiSteady' (=1 se il nodo è in steady state, altrimenti =0)
A = [[I; T],[0 0; E, D]];
vettToNodo = sum(A(1:N+1, 1:N));
vettFromNodo = sum(A(2:N+1, 1:N+2)');
vettNodiSteady = (vettToNodo==vettFromNodo);

% Choose default command line output for finestraInfoGrafo
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);


% --- Outputs from this function are returned to the command line.
function varargout = finestraInfoGrafo_OutputFcn(hObject, eventdata, handles)
% Get default command line output from handles structure
varargout{1} = handles.output;


function editVersoNodo_Callback(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.
function editVersoNodo_CreateFcn(hObject, eventdata, handles)
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in buttonApplica.
function buttonApplica_Callback(hObject, eventdata, handles)
global fig;
global C;

%leggo nel popup 'popupCatNodo' la categoria selezionata (es. Retail) e la
%salvo nella var 'catNodo':
value = get(handles.popupCatNodo, 'Value');
stringa = get(handles.popupCatNodo, 'String');
catNodo = stringa{value};
%salvo in 'nodoSel' il nodo selezionato dal 'popupNodoSel'
nodoSel = get(handles.popupNodoSel, 'Value') - 1;
%a seconda che nel bottone 'buttonApplica' ci sia la scritta 'Applica' o la
%scritta 'Aggiungi', eseguo due procedure diverse:
strButton = get(handles.buttonApplica, 'String');
if strcmp(strButton, 'Applica')
    if get(handles.cbImmagine, 'Value') == 1
        %controllo che esista l'immagine .jpg associata alla categoria
        %selezionata: essa deve trovarsi all'interno della cartella 'icone'
        %e avere lo stesso nome con cui è definita nel 'popupCatNodo':
        %se esiste eseguo la procedura per inserire l'icona nel grafo;
        %se non esiste visualizzo un mess di errore
        oldFolder = pwd;
        cd ..;
        percorsoIcona = pwd;
        percorsoIcona = strcat(percorsoIcona, '\icone+popup\', ...
            catNodo, '.jpg');
        cd(oldFolder);
        if exist(percorsoIcona, 'file')
            %richiamo la figura con il grafo:
            figure(fig);
            %leggo le coord del centro del cerchio nel grafo di 'nodoSel':
            coordCentro = C{1, nodoSel};
            %stabilisco le coordinate dell'immagine che vado ad inserire al
            %centro del cerchio
            %l'altezza dell'immagine è 3/4 risp la base x
            rapp = 3/4;
            x = sqrt(16*4/25);
            xA = coordCentro(1) - x/2;
            xB = coordCentro(1) + x/2;
            yA = coordCentro(2) - rapp*x/2;
            yD = coordCentro(2) + rapp*x/2;
            %carico la rispettiva icona
            imm = imread(percorsoIcona);
            %inserisco l'immagine nelle coord calcolate precedentemente
            hold on
            imagesc([xA xB], [yD yA], imm);
        else
            messaggio = strcat('file ''', percorsoIcona, ''' not found');
```

```
            msg = msgbox(messaggio, 'Error');
            ah = get( msg, 'CurrentAxes' );
            ch = get( ah, 'Children' );
            set( ch, 'FontSize', 8, 'HorizontalAlignment', 'left' );
        end;
    end;
    %aggiorno l'edit 'editLabel' con le info inserite dall'utente:
    label = strcat(num2str(nodoSel), '. ', catNodo);
    set(handles.editLabel, 'String', label);
else
    %se nel button 'buttonApplica' c'è la scritta 'Aggiungi' allora devo
    %inserire la nuova categoria inserita dall'utente nel popup
    %'popupCatNodo' e aggiornare il file 'popupCatNodo.txt'.
    %memorizzo nella stringa 'categoria' la categoria inserita dall'utente
    categoria = get(handles.editAltro, 'String');
    %aggiungo nella cella 'stringa' la categoria inserita dall'utente e
    %aggiorno il popup 'popupCatNodo':
    stringa{value} = categoria;
    stringa{value+1} = '..altro..';
    set(handles.popupCatNodo, 'String', stringa);
    %imposto la visualizzazione di alcuni oggetti:
    set(handles.editAltro, 'Visible', 'off');
    set(handles.cbImmagine, 'Visible', 'on');
    set(handles.buttonApplica, 'String', 'Applica');
    %salvo nel file 'popupCatNodo.txt' la stringa 'stringaCat' con le
    %categorie aggiornate:
    oldFolder = pwd;
    cd ..;
    percorsoFile = pwd;
    percorsoFile = strcat(percorsoFile, '\icone+popup\popupCatNodo.txt');
    cd(oldFolder);
    %apro il file in modalità di scrittura
    file = fopen(percorsoFile, 'wt');
    %memorizzo nella stringa 'stringaCat' tutte le categorie che ora si
    %trovano nell'array di celle 'stringa':
    nCategorie = length(stringa)-1;
    stringaCat = '';
    for cont = 1:nCategorie
        stringaCat = strcat(stringaCat, stringa{cont}, '\n');
    end;
    stringaCat = strcat(stringaCat, '..altro..');
    %scrivo nel file il contenuto di 'stringaCat'
    fprintf(file, stringaCat);
    %chiudo il file
    fclose(file);
end;


% --- Executes on selection change in popupCatNodo.
function popupCatNodo_Callback(hObject, eventdata, handles)

%leggo nel popup 'popupCatNodo' la categoria selezionata (es. Retail) e
%la salvo nella stringa 'catNodo':
value = get(handles.popupCatNodo, 'Value');
stringa = get(handles.popupCatNodo, 'String');
```

```
catNodo = stringa{value};
%a seconda di quello contenuto in 'catNodo' imposto la visualizzazione di
%alcuni oggetti:
if strcmp(catNodo, '..altro..')
    set(handles.editAltro, 'Visible', 'on');
    set(handles.cbImmagine, 'Value', 0);
    set(handles.cbImmagine, 'Visible', 'off');
    set(handles.buttonApplica, 'String', 'Aggiungi');
else
    set(handles.editAltro, 'Visible', 'off');
    set(handles.cbImmagine, 'Value', 1);
    set(handles.cbImmagine, 'Visible', 'on');
    set(handles.buttonApplica, 'String', 'Applica');
end;


% --- Executes during object creation, after setting all properties.
function popupCatNodo_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes during object creation, after setting all properties.
function finestraInfoGrafo_CreateFcn(hObject, eventdata, handles)

% Choose default command line output for finestraInfoGrafo
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);


% --- Executes on button press in buttonNodoSel.
function buttonNodoSel_Callback(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.
function popupNodoSel_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in popupNodoSel.
function popupNodoSel_Callback(hObject, eventdata, handles)
global C;
global fig;
global nodoSelPrec;
global vettNodiSteady;

%pulisco la label 'editLabel'
```

```
label = '';
set(handles.editLabel, 'String', label);
%salvo nella var 'nodoSel' il nodo selezionato dal popup 'popupNodoSel':
nodoSel = get(handles.popupNodoSel, 'Value') - 1;
if nodoSel ~= 0
    %abilito gli oggetti contenuti nel panel 'panelFlussi'
    ObjPanelFlussi = [handles.textVersoNodo, handles.textImport, ...
        handles.textExport, handles.textDissipato, ...
        handles.popupVersoNodo, handles.text2punti, ...
        handles.textFlussoVersoNodo, handles.textFlussoImport, ...
        handles.textFlussoExport, handles.textFlussoDissipato];
    set(ObjPanelFlussi, 'Enable', 'on');
    %visualizzo il panel 'panelInfoNodo'
    set(handles.panelInfoNodo, 'Visible', 'on');
    %scrivo i successori di 'nodoSel' nel popup 'popupVersoNodo':
    stringaSucc = ' ';
    nSucc = length(C{2, nodoSel});
    %se 'nodoSel' ha dei successori allora li vado a leggere dalla 2° riga
    %della matrice C e li salvo nella stringa 'StringaSucc',
    %intervallandoli da '|':
    if nSucc > 0
        stringaSucc = num2str(C{2, nodoSel}, '%d|');
        stringaSucc( stringaSucc == ' ' ) = [];
        stringaSucc( length(stringaSucc) ) = [];
    end;
    %importo i successori di 'nodoSel' nel popup 'popupVersoNodo'
    set(handles.popupVersoNodo, 'String', stringaSucc);
    set(handles.popupVersoNodo, 'Value', 1);
    if nSucc == 0
        set(handles.popupVersoNodo, 'Enable', 'off');
    elseif nSucc == 1
        set(handles.popupVersoNodo, 'Enable', 'inactive');
    else
        set(handles.popupVersoNodo, 'Enable', 'on');
    end;
    %richiamo la funzione 'popupVersoNodo_Callback':
    popupVersoNodo_Callback(handles.popupVersoNodo, eventdata, handles);
    %dalla 4°riga della matrice C leggo il flusso in IMPORT e lo visualizzo
    %nel text 'textFlussoImport'
    strFlusso = num2str( C{4, nodoSel} );
    set(handles.textFlussoImport, 'String', strFlusso);
    %dalla 5°riga della matrice C leggo il flusso in EXPORT e lo visualizzo
    %nel text 'textFlussoExport'
    strFlusso = num2str( C{5, nodoSel} );
    set(handles.textFlussoExport, 'String', strFlusso);
    %dalla 6°riga della matrice C leggo il flusso DISSIPATO e lo visualizzo
    %nel text 'textFlussoDissipato'
    strFlusso = num2str( C{6, nodoSel} );
    set(handles.textFlussoDissipato, 'String', strFlusso);

    %evidenzio di magenta il nodo selezionato, richiamando la funzione
    %'circle'; prima devo vedere se è selezionato un nodo ed eventualmente
    %colorarlo di nero (perchè è stato deselezionato):
    figure(fig);
    hold on;
```

```
        raggio = 1;
        %la var 'nodoSelPrec' contiene il nodo precedentemente selezionato
        if nodoSelPrec ~= 0
            circle(C{1,nodoSelPrec}, raggio);
        end;
        nodoSelPrec = nodoSel;
        colore = 'r';
        circle(C{1,nodoSel}, raggio, colore);
        hold off;

        %guardo se il nodo selezionato è in 'steady state' e applico la
        %rispettiva immagine nella finestra
        oldFolder = pwd;
        cd ..;
        percorsoImm = pwd;
        cd(oldFolder);
        if vettNodiSteady(nodoSel)
            percorsoImm = strcat(percorsoImm, '\icone+popup\STEADY STATE.jpg');
            imm = imread(percorsoImm);
        else
            percorsoImm = strcat(percorsoImm, '\icone+popup\NO STEADY STATE.jpg');
            imm = imread(percorsoImm);
        end;
        axes(handles.axesSteady);
        imagesc([0 1], [0 1], imm);
        axis off;
    else
        %imposto la visualizzazione degli oggetti nel panel 'panelFlussi':
        set(handles.textFlussoVersoNodo, 'String', ' ');
        set(handles.textFlussoImport, 'String', ' ');
        set(handles.textFlussoExport, 'String', ' ');
        set(handles.textFlussoDissipato, 'String', ' ');
        set(handles.popupVersoNodo, 'String', ' ');
        ObjPanelFlussi = [handles.textVersoNodo, handles.textImport, ...
            handles.textExport, handles.textDissipato, ...
            handles.popupVersoNodo, handles.text2punti, ...
            handles.textFlussoVersoNodo, handles.textFlussoImport, ...
            handles.textFlussoExport, handles.textFlussoDissipato];
        set(ObjPanelFlussi, 'Enable', 'off');
        %imposto la visualizzazione del panel 'panelInfoNodo':
        set(handles.panelInfoNodo, 'Visible', 'off');
        %deseleziono dal grafo il cerchio precedentemente evidenziato
        if nodoSelPrec ~= 0
            figure(fig);
            hold on;
            raggio = 1;
            circle(C{1,nodoSelPrec}, raggio);
            nodoSelPrec = 0;
            hold off;
        end;
        axes(handles.axesSteady);
        cla;
    end;
```

```
% --- Executes on selection change in popupVersoNodo.
function popupVersoNodo_Callback(hObject, eventdata, handles)
global C;

%leggo il nodo selezionato 'nodoSel' e il nodo 'nodoTo' verso il quale si
%vuole conoscere il flusso:
nodoSel = get(handles.popupNodoSel, 'Value') - 1;
if (~isempty(C{2, nodoSel}))
    valueTo = get(handles.popupVersoNodo, 'Value');
    %dalla riga 3 della matrice C leggo il flusso da 'nodoSel' a 'nodoTo' e
    %lo salvo nella stringa 'strFlusso':
    flusso = C{3, nodoSel}(1, valueTo);
    strFlusso = num2str( flusso );
else
    strFlusso = ' ';
end
%scrivo il flusso 'strFlusso' nel text 'textFlussoVersoNodo'
set(handles.textFlussoVersoNodo, 'String', strFlusso);


% --- Executes during object creation, after setting all properties.
function popupVersoNodo_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function editAltro_Callback(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.
function editAltro_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in cbImmagine.
function cbImmagine_Callback(hObject, eventdata, handles)


function editLabel_Callback(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.
function editLabel_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on button press in buttonInserisci.
function buttonInserisci_Callback(hObject, eventdata, handles)
global fig;

label = get(handles.editLabel, 'String');
figure(fig);
hold on;
gtext(label);
hold off;


% --- Executes when user attempts to close finestraInfoGrafo.
function finestraInfoGrafo_CloseRequestFcn(hObject, eventdata, handles)
global nodoSelPrec;
global C;
global fig;

%la var 'nodoSelPrec' contiene il nodo precedentemente selezionato
if nodoSelPrec ~= 0
    figure(fig);
    hold on;
    raggio = 1;
    circle(C{1,nodoSelPrec}, raggio);
    hold off;
end;


% Hint: delete(hObject) closes the figure
delete(hObject);


% --- Executes on button press in buttonInserisciFlussi.
function buttonInserisciFlussi_Callback(hObject, eventdata, handles)
global fig;
global C;
global T;
global I;
global E;
global D;
global spazioLivelli;
global labelNum;

N = length(I);
figure(fig);
hold on;

%leggo la scritta presente nel bottone 'buttonInserisciFlussi'
buttonStringa = get(handles.buttonInserisciFlussi, 'String');
%se il bottone premuto ha scritta 'Visualizza Flussi' allora eseguo la
%procedura per visualizzare i flussi nel grafo, altrimenti li cancello
if strcmp(buttonStringa, 'Visualizza Flussi')
%leggo la dimensione dei numeri impostata attraverso 'sliderDimNum'
DimNum = get(handles.sliderDimNum, 'Value');
```

```
%procedura per visualizzare i flussi nel grafo:
raggio = 1;
xmax = C{7,1}(1,1);
ymax = C{7,1}(1,2);
labelNum = [];
contatore = 0;
for cont1 = 1:N
    %flussi interni
    C1 = C{1,cont1};
    nfrecce = length(C{2,cont1});
    for cont2 = 1 : nfrecce
        C2 = C{1, C{2,cont1}(1,cont2)};
        [A,B] = estremita(C1, C2, raggio);
        PMedio = [];
        if ( T(C{2,cont1}(1,cont2), cont1) ~= 0 ) || ( A(1)==B(1) ) || ...
                ( A(2)==B(2) )
            AB = sqrt( (A(1)-B(1))^2 +(A(2)-B(2))^2 );
            AO = AB;
            AC = AB/2;
            OC = sqrt( (AO^2) - (AC^2) );
            PC = AO-OC;
            Cy = (A(2)+B(2))/2;
            Cx = (A(1)+B(1))/2;
            alfa = atan2( (B(1)-A(1)), (B(2)-A(2)) );
            %trovo il punto medio
            PMedio(1) = Cx + PC*cos(alfa);
            PMedio(2) = Cy + PC*sin(alfa);
        else
            %trovo il punto medio
            PMedio(1) = (C1(1)+C2(1))/2;
            PMedio(2) = (C1(2)+C2(2))/2;
        end;
        %scrivo i flussi interni
        flusso = T(cont1, C{2,cont1}(1,cont2));
        contatore = contatore+1;
        labelNum(contatore) = text(PMedio(1), PMedio(2), num2str(flusso), ...
            'FontSize', DimNum);
    end;
    %'Import'
    if C{4,cont1} ~= 0
        PMedio = [];
        h1 = ymax - C{1,cont1}(1,2);
        h2 = C{1,cont1}(1,2);
        if C{1,cont1}(1,1) <= spazioLivelli*1.8
            yf = C{1,cont1}(1,2);
            xf = 0;
        elseif h1<h2
            yf = ymax;
            xf = C{1,cont1}(1,1) - spazioLivelli*( 0.3 + (h1/(ymax/2)) );
        else
            yf = 0;
            xf = C{1,cont1}(1,1) - spazioLivelli*( 0.3 + (h2/(ymax/2)) );
        end;
        [~,B] = estremita([xf, yf], C1, raggio);
        %trovo il punto medio
```

```
            PMedio(1) = (B(1)+xf)/2;
            PMedio(2) = (B(2)+yf)/2;
            %scrivo i flussi in import
            flusso = I(1, cont1);
            contatore = contatore+1;
            labelNum(contatore) = text(PMedio(1), PMedio(2), num2str(flusso), ...
                'FontSize', DimNum);
        end;
        %disegno le frecce in 'Export'
        if C{5,cont1} ~= 0
            PMedio = [];
            h1 = ymax - C{1,cont1}(1,2);
            h2 = C{1,cont1}(1,2);
            if C{1,cont1}(1,1) >= xmax-spazioLivelli*1.8
                yf = C{1,cont1}(1,2);
                xf = xmax;
            elseif h1 < h2
                yf = ymax;
                xf = C{1,cont1}(1,1) + spazioLivelli*( 0.3 + (h1/(ymax/2)) );
            else
                yf = 0;
                xf = C{1,cont1}(1,1) + spazioLivelli*( 0.3 + (h2/(ymax/2)) );
            end;
            [~,B] = estremita([xf, yf], C1, raggio);
            %trovo il punto medio
            PMedio(1) = (B(1)+xf)/2;
            PMedio(2) = (B(2)+yf)/2;
            %scrivo i flussi in export
            flusso = E(cont1, 1);
            contatore = contatore+1;
            labelNum(contatore) = text(PMedio(1), PMedio(2), num2str(flusso), ...
                'FontSize', DimNum);
        end;
        %'Dissipation'
        if C{6,cont1} ~= 0
            %trovo il punto medio
            xd = C{1,cont1}(1,1) + raggio* sqrt(2)/2;
            yd = C{1,cont1}(1,2) - raggio* sqrt(2)/2;
            %scrivo i flussi dissipati
            flusso = D(cont1, 1);
            contatore = contatore+1;
            labelNum(contatore) = text(xd+0.36, yd-0.5, num2str(flusso), ...
                'FontSize', DimNum);
        end;
    end;
    %aggiorno la scritta nel bottone 'buttonInserisciFlussi'
    set(handles.buttonInserisciFlussi, 'String', 'Nascondi Flussi');
    else
    %cancello i numeri riportanti i flussi dal grafo
    delete(labelNum(:));
    %aggiorno la scritta nel bottone 'buttonInserisciFlussi'
    set(handles.buttonInserisciFlussi, 'String', 'Visualizza Flussi');
    end;
    hold off;
```

```
% --- Executes on slider movement.
function sliderDimNum_Callback(hObject, eventdata, handles)

%leggo la dimensione dei caratteri impostata attraverso 'sliderDimNum'
sliderValue = get(handles.sliderDimNum, 'Value');
sliderValue = round(sliderValue);
set(handles.sliderDimNum, 'Value', sliderValue);
%cambio la dimensione dei caratteri, aggiornando i flussi nel grafo se sono
%visualizzati
buttonStringa = get(handles.buttonInserisciFlussi, 'String');
if ~strcmp(buttonStringa, 'Visualizza Flussi')
    buttonInserisciFlussi_Callback(hObject, eventdata, handles);
    buttonInserisciFlussi_Callback(hObject, eventdata, handles);
end;


% --- Executes during object creation, after setting all properties.
function sliderDimNum_CreateFcn(hObject, eventdata, handles)

if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end
```

```
function varargout = finestraModificaGrafo(varargin)
% FINESTRAMODIFICAGRAFO M-file for finestraModificaGrafo.fig
% Last Modified by GUIDE v2.5 25-Aug-2011 10:37:09
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @finestraModificaGrafo_OpeningFcn, ...
                   'gui_OutputFcn',  @finestraModificaGrafo_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before finestraModificaGrafo is made visible.
function finestraModificaGrafo_OpeningFcn(hObject, eventdata, handles,
varargin)
global spazioLivAttuale;

%posiziono la finestra nello schermo
set(handles.finestraModificaGrafo, 'Units', 'Normalized', 'Position', [0.66
0.5 0.315 0.5]);
%imposto le proprietà di slider 'sliderSpazioLivelli' che va da un valore
%(distanza) minimo di 3 ad un max di 9
slider_step(1:2) = 1/(9-3);
set(handles.sliderSpazioLivelli,'sliderstep',slider_step,
'max',9,'min',3,'Value',4)
spazioLivAttuale = 4;

% Choose default command line output for finestraModificaGrafo
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);


% --- Outputs from this function are returned to the command line.
function varargout = finestraModificaGrafo_OutputFcn(hObject, eventdata,
handles)
% Get default command line output from handles structure
varargout{1} = handles.output;


function editNodo_Callback(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.
function editNodo_CreateFcn(hObject, eventdata, handles)
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in buttonSpostaNodo.
function buttonSpostaNodo_Callback(hObject, eventdata, handles)
global fig;
global C;

[~, N] = size(C);
Nodo = get(handles.editNodo, 'String');
if (strcmp(Nodo, '')) || (str2num(Nodo) > N) || (str2num(Nodo) < 1)
    msg = msgbox('nodo non trovato', 'Error', 'error');
else
    %seleziono la figura avente indirizzo 'fig':
    figure(fig);
    %memorizzo in 'newCoord' le coordinate del punto selezionato dall'utente:
    newCoord = [ginput(1)];
    %leggo dall'edit 'editNodo' di 'finestraImpostazioni' il nodo da
    %spostare e lo converto in numero:
    nodo = str2num( get(handles.editNodo, 'String') );
    %pulisco la finestra della figura:
    clf(fig);
    C{1,nodo} = newCoord;
    creaGrafico(C);
end;


% --- Executes on slider movement.
function sliderSpazioLivelli_Callback(hObject, eventdata, handles)
global fig;
global C;
global spazioLivelli;
global spazioLivAttuale;

%leggo il valore impostato tramite 'sliderSpazioLivelli' e rifaccio il
%grafo:
sliderValue = get(handles.sliderSpazioLivelli, 'Value');
sliderValue = round(sliderValue);
set(handles.sliderSpazioLivelli, 'Value', sliderValue);
%seleziono la figura avente indirizzo 'fig':
figure(fig);
%pulisco la finestra della figura:
clf(fig);
spazioLivelli = sliderValue;
[~, N] = size(C);
for nodo = 1:N
    x = C{1,nodo}(1,1);
    livelloNodo = x/spazioLivAttuale;
    x = livelloNodo*spazioLivelli;
    C{1,nodo}(1,1) = x;
end;
```

```
xmax = C{7,1}(1,1) * (spazioLivelli/spazioLivAttuale);
C{7,1}(1,1) = xmax;
spazioLivAttuale = spazioLivelli;
creaGrafico(C);


% --- Executes during object creation, after setting all properties.
function sliderSpazioLivelli_CreateFcn(hObject, eventdata, handles)

if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on button press in buttonConferma.
function buttonConferma_Callback(hObject, eventdata, handles)
global hSchermataPrincipale;

finestraModificaGrafo_CloseRequestFcn(finestraModificaGrafo);
set(hSchermataPrincipale, 'Visible', 'on');
finestraInfoGrafo();


% --- Executes when user attempts to close finestraModificaGrafo.
function finestraModificaGrafo_CloseRequestFcn(hObject, eventdata, handles)

% Hint: delete(hObject) closes the figure
delete(hObject);


% --- Executes on button press in buttonRifaiGrafo.
function buttonRifaiGrafo_Callback(hObject, eventdata, handles)
global T;
global I;
global E;
global D;
global fig;

scelta = questdlg('Verrà modificata la matrice originale. Continuare?', ...
    'question dialog', 'Sì', 'No', 'No');
switch scelta
    case 'No'
        %torno alla schermata senza fare nulla
    case 'Sì'
        %richiamo la funzione 'ordinaMatrici' che prova a ordinare la
        %matrice iniziale in modo da ottenere un grafo migliore
        [ T, I, E, D, newNumerazione ] = ordinaMatrici(T, I, E, D);
        [C] = creaMatriceC (T, I, E, D);
        figure(fig);
        clf(fig);
        creaGrafico(C);
        msgbox('La nuova matrice è stata salvata nella cartella ...
            ''risultati''', 'msgbox');
end
```

```
function varargout = finestraPercorsoFile(varargin)
% FINESTRAPERCORSOFILE M-file for finestraPercorsoFile.fig
% Last Modified by GUIDE v2.5 17-Jul-2011 13:37:43
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @finestraPercorsoFile_OpeningFcn, ...
                   'gui_OutputFcn',  @finestraPercorsoFile_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before finestraPercorsoFile is made visible.
function finestraPercorsoFile_OpeningFcn(hObject, eventdata, handles,
varargin)

% Choose default command line output for finestraPercorsoFile
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);


% --- Outputs from this function are returned to the command line.
function varargout = finestraPercorsoFile_OutputFcn(hObject, eventdata,
handles)

% Get default command line output from handles structure
varargout{1} = handles.output;


function editPercorsoFile_Callback(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.
function editPercorsoFile_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in buttonOk.
function buttonOk_Callback(hObject, eventdata, handles)
```

```
global percorsoFile;
global hSchermataPrincipale;

%premendo il bottone 'Ok' della finestra 'finestraPercorsoFile', prima di
%lanciare il programma per creare il grafico, controllo se esiste il file
%con il percorso inserito: se non lo trovo, avviso l'utente con un msg
percorsoFile = '';
percorsoFile = get(handles.editPercorsoFile, 'String');
if exist(percorsoFile, 'file')
    finestraPercorsoFile_CloseRequestFcn(finestraPercorsoFile);
    hSchermataPrincipale = schermataPrincipale();
    finestraModificaGrafo();
else
    msg = msgbox('      file not found', 'Error');
    ah = get( msg, 'CurrentAxes' );
    ch = get( ah, 'Children' );
    set( ch, 'FontSize', 12, 'HorizontalAlignment', 'left' );
end;


% --- Executes on button press in buttonBrowse.
function buttonBrowse_Callback(hObject, eventdata, handles)

%premendo il bottone 'Browse..' si ha la possibilità di cercare un file
%(avente estensione tipica di un foglio di calcolo) tra le directory
[fileName, path] = uigetfile({'*.xls; *.xlsx', 'Excel Files (*.xls, *.xlsx)';
'*.*', 'All Files (*.*)'}, 'Seleziona il file con la matrice flussi');
if ~isequal(fileName,0)
    percorsoFile = '';
    percorsoFile = strcat(path, fileName);
    set(handles.editPercorsoFile, 'String', percorsoFile);
end;


% --- Executes when user attempts to close finestraPercorsoFile.
function finestraPercorsoFile_CloseRequestFcn(hObject, eventdata, handles)

% Hint: delete(hObject) closes the figure
delete(hObject);
```

```
function [ G, L, CI, prob, perc ] = indiciCicli ( T, I, E, D, aCicli )
%da questa funzione ottengo la matrice G, la matrice L (matrice di
%Leontief), TST, CI (indice di Finn), le matrici LC e LNC, la probabilità
%di ogni ciclo e le rispettive percentuali

%trovo il numero di nodi N:
N = length(T);
%creo la matrice G
S = sum(T) + I;
G = [];
for cont=1:N
    G = [G; T(cont,:)./S];
end;
%G

%creo la matrice L (matrice di Leontief)
Id = eye(size(G));
L = [Id-G]^(-1);
[Id-G]
%L
%calcolo l'indice di Finn CI e creo le matrici C, LN, LNC (L=LC+LNC)
TST = sum(sum(T)) + sum(I) + sum(E) + sum(D);
CI = 0;
for cont = 1:N
    CI = CI+ ( (S(cont)/TST) * ((L(cont,cont)-1)/L(cont,cont)) );
end;
C = zeros(size(L));
for cont = 1:N
    C(cont,cont) = (L(cont,cont)-1)/L(cont,cont);
end;
%C
LC = L*C
LNC = L-LC

%richiamo la funzione 'arrayCicli' che mi restituisce l'array di celle
%'aCicli' contenente tutti i cicli salvati (in diverse celle) sottoforma di
%array numerici
%(se in input alla funzione 'indiciCicli' do già la var 'aCicli' allora non
%richiamo la funzione 'arrayCicli'!)
if nargin == 4
    [ aCicli, ~ ] = arrayCicli(T);
end;

%calcolo la probabilità di ogni ciclo moltiplicando i coeff della matrice G
%che appartengono al percorso/ciclo; tali probabilità le salvo nell'array
%'prob'
ncelle = length(aCicli);
prob(1:ncelle) = 1;
for cont = 1:ncelle
    prec = 0;
    array = (aCicli{1, cont})';
    for ind = array
        if prec ~= 0
            prob(cont) = prob(cont) * G(prec, ind);
        end;
```

```
        prec = ind;
    end;
end;
%calcolo la percentuale  di probabilità di ogni ciclo, pesando la sua
%probabilità sul totale, somma delle prob di tutti i cicli; salvo tale
%percentuali nell'array 'perc'
perc(1:ncelle) = 0;
somma = sum(prob);
for cont = 1:ncelle
    perc(cont) = (prob(1, cont) / somma) * 100;
end;

end
```

```
function [ TST, AMI, C, A, OI, OE, OD, R, A_C, OI_C, OE_C, OD_C, R_C  ] =
    indiciEntropici ( T )

%trovo TST=t..
TST = sum((sum(T))');
%trovo vettore somma colonne t.j
Tj = sum(T);
%trovo vettore somma righe t.i
Ti = sum(T');
%trovo dimensione matrice globale
[~,m]=size(T);
%trovo num. partner rete (dimensione matr M)
n=m-3;
%identifico M(matrice flussi solo interni al sistema)
M=T(2:n+1, 2:n+1);
%trovo vettore somma colonne m.j
Mj = sum(M);
%trovo vettore somma righe mi.
Mi = sum(M');
%identifico vettore INPUT
INPUT = T(1,2:n+1);
%identifico vettore OUTPUT
OUTPUT = T(2:n+1,n+2);
%identifico vettore DISSIPATION
DISSIPATION = T(2:n+1,n+3);

%calcolo matrice delle entropie in output Ho
Ho=zeros(m,1);
for i=1:m
    if Ti(i)==0;
        Ho(i)=0;
    else
        Ho(i)=-((Ti(i)/TST)*log2(Ti(i)/TST));
    end
end

%calcolo entropia di output Hout
Hout=sum(Ho)

%calcolo matrice delle entropie in input Hi
Hi=zeros(m,1);
for j=1:m
    if Tj(j)==0;
        Hi(j)=0;
    else
        Hi(j)=-((Tj(j)/TST)*log2(Tj(j)/TST));
    end
end

%calcolo entropia di input Hin
Hin=sum(Hi)

%calcolo matrice delle entropie congiunte Hio
Hio=zeros(m,m);
for i=1:m
```

```
    for j=1:m
    if T(i,j)==0;
        Hio(i,j)=0;
    else
        Hio(i,j)=-((T(i,j)/TST)*log2(T(i,j)/TST));
    end
    end
end
Hio
%calcolo entropia congiunta HIO
HIO = sum((sum(Hio))')

%calcolo matrice delle entropie condizionali Hi/o
HiCo=zeros(m,m);
for i=1:m
    for j=1:m
    if T(i,j)==0;
        HiCo(i,j)=0;
    else
        HiCo(i,j)=-((T(i,j)/TST)*log2(T(i,j)/Ti(i)));
    end
    end
end

%calcolo entropia condizionale HI/O
HIcO = sum((sum(HiCo))');

%calcolo matrice delle entropie condizionali Ho/i
HoCi=zeros(m,m);
for i=1:m
    for j=1:m
    if T(i,j)==0;
        HoCi(i,j)=0;
    else
        HoCi(i,j)=-((T(i,j)/TST)*log2(T(i,j)/Tj(j)));
    end
    end
end

%calcolo entropia condizionale HO/I
HOcI = sum((sum(HoCi))');

%calcolo indici entropici
AMI=Hout+Hin-HIO
C=HIO*TST;
A=AMI*TST;
O=C-A;

%calcolo matrice per overhead da input
Oi=zeros(1,n);
for j=1:n
    if T(1,j+1)==0
        Oi(j)=0;
    elseif Tj(j+1)==0
        Oi(j)=0;
```

```
    else
        Oi(j)=T(1,j+1)*(log2((T(1,j+1))^2/(Tj(j+1)*(sum(INPUT)))));
    end
end
OI=-(sum(Oi));

%calcolo matrice per overhead da export
Oe=zeros(1,n);
for i=1:n
    if OUTPUT(i)==0
        Oe(i)=0;
    elseif (sum(OUTPUT)*Ti(i+1))==0
        Oe(i)=0;
    else
        Oe(i)=OUTPUT(i)*log2((OUTPUT(i))^2/(sum(OUTPUT)*Ti(i+1)));
    end
end
OE=-(sum(Oe));

%calcolo matrice per overhead da dissipation
Od=zeros(1,n);
for i=1:n
    if DISSIPATION(i)==0
        Od(i)=0;
    elseif (sum(DISSIPATION)*Ti(i+1))==0
        Od(i)=0;
    else

Od(i)=DISSIPATION(i)*log2((DISSIPATION(i))^2/(sum(DISSIPATION)*Ti(i+1)));
    end
end
OD=-(sum(Od));

%calcolo matrice per overhead da redundancy
Or=zeros(n);
for i=1:n
    for j=1:n
        if M(i,j)==0
        Or(i,j)=0;
        elseif (Tj(j+1)*Ti(i+1))==0
        Or(i,j)=0;
        else
        Or(i,j)=M(i,j)*log2((M(i,j))^2/(Tj(j+1)*Ti(i+1)));
        end
    end
end
R=-(sum((sum(Or))'));

%Percentage:
A_C=(A/C)*100;
OI_C=(OI/C)*100;
OE_C=(OE/C)*100;
OD_C=(OD/C)*100;
R_C=(R/C)*100;
end
```

```
function [ T, I, E, D, newNumerazione ] = ordinaMatrici ( T, I, E, D )
%questa funzione ha lo scopo di ordinare la matrice con i flussi interni
%(matrice T) per ottenere successivamente un grafico ordinato e
%rappresentativo della supply chain;
%la funzione si articola in 3 parti: la prima sposta i nodi dai quali non
%arrivano frecce da altri nodi, a sx della matrice (verosimilmente saranno
%i fornitori); la seconda parte ordina i nodi in base ai successori; la
%terza ordina i nodi spaiati sempre in base ai successori.

%1° PARTE
%trovo il numero di nodi N:
N = length(T);
numerazione = [1:N];
%sposto a sx della matrice T le colonne la cui somma è =0 (significa che a
%quei nodi non arrivano frecce da altri nodi)
sommaColonna = sum(T);
%in 'arrayColonneDaSpostare' salvo gli indici delle colonne da spostare
arrayColonneDaSpostare = find(sommaColonna == 0);
%'indiceT' è la variabile che tiene conto a che nodo della matrice T sono
%arrivato:
indiceT = 0;
%dato che spostando le colonne (e le righe) della matrice vado a mischiare
%l'ordine dei nodi, in 'newNumerazione' salvo il nuovo ordine
newNumerazione = [1:N];

if ~isempty(arrayColonneDaSpostare)
    %richiamo la funzione 'spostaColonne' che, data in input una matrice o
    %la numerazione da aggiornare, sposta le colonne
    %'arrayColonneDaSpostare' nel punto della matrice indicato da 'indiceT'
    [T, ~] = spostaColonne(T, indiceT, arrayColonneDaSpostare);
    %sposto anche le righe usando sempre la funzione 'spostaColonne', dopo
    %aver fatto la trasposta di T:
    T2=T';
    [T2, ~] = spostaColonne(T2, indiceT, arrayColonneDaSpostare);
    T=T2';
    %aggiorno l'ordine della numerazione 'newNumerazione'
    [newNumerazione, indiceT] = spostaColonne(newNumerazione, indiceT, ...
        arrayColonneDaSpostare);

    %2° PARTE
    %esploro le righe da 'prec' a 'succ' per trovarne i successori (ovvero
    %i nodi colonna con valore >0) e li sposto a sx della matrice, dopo
    %'indiceT'
    prec = 1;
    succ = indiceT;
    %continuo a iterare la procedura finchè trovo dei successori
    trovaSucc = true;
    while trovaSucc
        trovaSucc = 0;
        for riga = prec : succ
            T1 = T;
            T1(:, 1:indiceT) = 0;
            arrayColonneDaSpostare = find(T1(riga, :)>0);
            %sposto le colonne:
            [T, ~] = spostaColonne(T, indiceT, arrayColonneDaSpostare);
```

```
            %sposto le righe:
            T2=T';
            [T2, ~] = spostaColonne(T2, indiceT, arrayColonneDaSpostare);
            T=T2';
            %aggiorno la numerazione:
            [newNumerazione, indiceT] = spostaColonne(newNumerazione, ...
                indiceT, arrayColonneDaSpostare);
            if ~isempty(arrayColonneDaSpostare)
                trovaSucc = 1;
            end;
        end;
        prec=succ+1;
        succ=indiceT;
    end;
end;

%3° PARTE
for cont = indiceT+1:N
    %trovo i successori del nodo 'cont'
    y = find(T(cont, :)>0);
    if (length(y)>0) && (cont>y(1))
        %sposto la colonna 'cont' prima del suo primo successore:
        [T, ~] = spostaColonne(T, y(1)-1, cont);
        %sposto la riga 'cont':
        T2=T';
        [T2, ~] = spostaColonne(T2, y(1)-1, cont);
        T=T2';
        %aggiorno la numerazione:
        [newNumerazione, ~] = spostaColonne(newNumerazione, y(1)-1, cont);
    end;
end;

%aggiorno le matrici di import I, export E, dissipation D
I1 = I;
E1 = E;
D1 = D;
for cont=1:N
    I(1, cont) = I1(1, newNumerazione(cont));
    E(cont, 1) = E1(newNumerazione(cont), 1);
    D(cont, 1) = D1(newNumerazione(cont), 1);
end;
MatriceOrdinata = cell(N+4, N+4);
MatriceOrdinata(:, 2) = {''};
MatriceOrdinata(N+3:N+4, :) = {''};
MatriceOrdinata(1, :) = [{''}, {'Imp.'}, num2cell(newNumerazione), ...
    {'Exp.'}, {'Diss.'}];
MatriceOrdinata(:, 1) = [{''}; {'Imp.'}; num2cell(newNumerazione'); ...
    {'Exp.'}; {'Diss.'}];
MatriceOrdinata(2, :) = [{'Imp.'}, {''}, num2cell(I), {''},{''}];
MatriceOrdinata(3:N+2, N+3) = [num2cell(E)];
MatriceOrdinata(3:N+2, N+4) = [num2cell(D)];
MatriceOrdinata(3:N+2, 3:N+2) = [num2cell(T)];

%salvo all'interno della cartella 'risultati' il file 'nuovaMatrice.xls'
%contenente la matrice appena creata
```

```
oldFolder = pwd;
cd ..;
percorsoFileNuovaMatrice = pwd;
percorsoFileNuovaMatrice = strcat(percorsoFileNuovaMatrice, ...
    '\risultati\nuovaMatrice.xls');
cd(oldFolder);
if exist(percorsoFileNuovaMatrice)
    warndlg('Attenzione, il file ''nuovaMatrice.xls'' sarà sovrascritto', ...
        'Attenzione');
    delete(percorsoFileNuovaMatrice);
end;
xlswrite(percorsoFileNuovaMatrice, MatriceOrdinata, 'Foglio1', 'A1');

end
```

```
function varargout = schermataPrincipale(varargin)
% SCHERMATAPRINCIPALE M-file for schermataPrincipale.fig
% Last Modified by GUIDE v2.5 17-Jul-2011 22:46:03
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @schermataPrincipale_OpeningFcn, ...
                   'gui_OutputFcn',  @schermataPrincipale_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before schermataPrincipale is made visible.
function schermataPrincipale_OpeningFcn(hObject, eventdata, handles, varargin)
global percorsoFile;
global aCicli;
global infoRicircolo;
global T;
global I;
global E;
global D;

%posiziono la finestra 'schermataPrincipale' nello schermo
set(handles.schermataPrincipale, 'Units', 'Normalized', ...
    'Position', [0.68 0.55 0.32 0.48]);
%creo le matrici T, I, E, D richiamando la funzione creaMatriciTIED dando in
%input il percorso del file inserito dall'utente:
[T, I, E, D] = creaMatriciTIED (percorsoFile);
%creo il grafo richiamando la funzione 'creaGrafico':
creaGrafico();
%inserisco nel popup-menù 'popupRicircolo' le stringhe con i ricircoli
%create a partire dall'array di celle 'aCicli' contenente tutti i cicli
%sottoforma di array numerici
%creo un'unica stringa 'string' contenente tutti i cicli separati da '|'
%che il popup interpreta come andar a capo:
string = 'nessuno|tutti';
ncelle = length(aCicli);
for cont = 1:ncelle
    string = strcat( string, '|' );
    %per ogni ciclo contenuto in 'aCicli' converto i valori in stringa e li
    %aggiungo a 'string', intervallandoli con '->':
    narr = length(aCicli{1, cont});
    for cont2 = 1:narr
        string = strcat( string, num2str(aCicli{1, cont}(cont2)), '->' );
```

```
    end;
    lunghstring = length(string);
    string((lunghstring-1):lunghstring) = '';
end;
%inserisco la stringa 'string' nel popup 'popupRicircolo':
set( handles.popupRicircolo, 'String', string );
%se non ci sono ricircoli allora disabilito il popup 'popupRicircolo'
if strcmp(string, 'nessuno|tutti')
    set( handles.popupRicircolo, 'Enable', 'off' );
else
    set( handles.popupRicircolo, 'Enable', 'on' );
end;
%creo l'array di celle 'infoRicircolo' nel quale andrò a salvare le info su
%ogni ricircolo inserite dall'utente (il tipo di ricircolo):
infoRicircolo = cell(1, ncelle);

%creo il popup 'popupCatRicircolo'.
%leggo le categorie dal file 'popupCatRicircolo.txt' (che deve trovarsi
%nella cartella 'icone+popup' che a sua volta deve trovarsi nella stessa
%directory della cartella 'funzioni')  e le carico nel popup
%'popupCatRicircolo'.
oldFolder = pwd;
cd ..;
percorsoFile = pwd;
percorsoFile = strcat(percorsoFile, '\icone+popup\popupCatRicircolo.txt');
cd(oldFolder);
%apro l'archivio in modalità di lettura:
file = fopen(percorsoFile, 'r');
%salvo nell'array di celle 'stringaTipiCiclo' il contenuto del file:
stringaTipiCiclo = textscan(file, '%s','delimiter', '\n');
stringaTipiCiclo = stringaTipiCiclo{1};
%carico nel 'popupCatNodo' le categorie contenute in 'stringaTipiCiclo':
set(handles.popupCatRicircolo, 'String', stringaTipiCiclo);
%chiudo l'archivio:
fclose(file);

% Choose default command line output for schermataPrincipale
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);


% --- Outputs from this function are returned to the command line.
function varargout = schermataPrincipale_OutputFcn(hObject, eventdata,
handles)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on selection change in popupRicircolo.
function popupRicircolo_Callback(hObject, eventdata, handles)
global infoRicircolo;
global aCicli;
global T;
```

```
global C;
global fig;

%leggo le info memorizzate in 'infoRicircolo' e le visualizzo a schermo:
value = get(handles.popupRicircolo, 'Value');
if (value ~= 1) && (value ~= 2)
    set( handles.buttonInserisciRicircolo, 'Enable', 'on' );
    tipo = infoRicircolo{1, value-2};
else
    set( handles.buttonInserisciRicircolo, 'Enable', 'off' );
    tipo = '';
end;
set( handles.textTipoRicircolo, 'String', tipo );

%procedura per evidenziare i cicli selezionati da 'popupRicircolo'
%richiamo la figura con il grafo:
figure(fig);
hold on;
%se nel 'popupRicircolo' viene selezionata la voce 'nessuno' allora
%'value'=1, se viene selezionata la voce 'tutti' allora 'value'=2, se viene
%selezionato un ricircolo 'value'>2
if value==1
    %se 'value'=1 pulisco il grafico, lo disegno richiamando la funzione
    %'creaGrafico' e riposiziono la legenda:
    legenda = legend();
    posLegenda = get(legenda, 'Position');
    clf(fig);
    creaGrafico(C);
    legenda = legend();
    set(legenda, 'Position', posLegenda);
elseif value>1
    %salvo nell'array 'cicliSel' i cicli da evidenziare: se 'value'=2
    %allora 'cicliSel' contiene tutti i cicli:
    if value==2
        nCicli = length(aCicli);
        cicliSel = [1:nCicli];
    else
        cicliSel = [value-2];
    end;
    %ripeto la seguente procedura per ogni ciclo contenuto in 'cicliSel'
    for indCiclo = [cicliSel]
        %'nFrecceCiclo' è pari al numero di collegamenti nel ciclo che sto
        %considerando (aCicli{1, indCiclo}):
        nFrecceCiclo = length(aCicli{1, indCiclo});
        %evidenzio di magenta i collegamenti tra i nodi del ciclo
        %considerato, individuando ad ogni iterazione una coppia di nodi da
        %collegare:
        for cont = 1:(nFrecceCiclo-1)
            %individuo la coppia di nodi da collegare:
            nodo1 = aCicli{1, indCiclo}(cont, 1);
            nodo2 = aCicli{1, indCiclo}(cont+1, 1);
            %il resto della procedura, che evidenzia i collegamenti tra i
            %nodi, è presa pari pari da quella implementata nella funzione
            %'creaGrafico'
            C1 = C{1, nodo1};
```

```
            C2 = C{1, nodo2};
            raggio = 1;
            [A,B] = estremita(C1, C2, raggio);
            spessore = 1.3;
            colore = 'm';
            if ( T(nodo2, nodo1) ~= 0 ) || ( A(1)==B(1) )
                AB = sqrt( (A(1)-B(1))^2 +(A(2)-B(2))^2 );
                AO = AB;
                AC = AB/2;
                OC = sqrt( (AO^2) - (AC^2) );
                PC = AO-OC;
                Cy = (A(2)+B(2))/2;
                Cx = (A(1)+B(1))/2;
                alfa = atan2( (B(1)-A(1)), (B(2)-A(2)) );
                P(1) = Cx + PC*cos(alfa);
                P(2) = Cy + PC*sin(alfa);
                xy = [];
                xy = [A(1), P(1), B(1); A(2), P(2), B(2)];
                t = 1:3;
                ts = 1: 0.1: 3;
                xys = spline(t,xy,ts);
                %plot the interpolated curve.
                plot(xys(1,:),xys(2,:), 'LineWidth', spessore, ...
                    'Color', colore);
                %disegno la punta della freccia:
                R(1) = B(1) + ((P(1)-B(1))/10000);
                R(2) = B(2) + ((P(2)-B(2))/10000);
                arrow(R, B, 8, 'BaseAngle', 60, 'LineWidth', spessore, ...
                    'FaceColor', colore, 'EdgeColor', colore);
            else
                arrow(A, B, 8, 'BaseAngle', 60, 'LineWidth', spessore, ...
                    'FaceColor', colore, 'EdgeColor', colore);
            end;
        end;
    end;
end;
hold off;


% --- Executes during object creation, after setting all properties.
function popupRicircolo_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in buttonInserisciRicircolo.
function buttonInserisciRicircolo_Callback(hObject, eventdata, handles)
set( handles.popupRicircolo, 'Enable', 'Inactive' );
set( handles.panelInfoRicircolo, 'Visible', 'off' );
set( handles.panelInserisciRicircolo, 'Visible', 'on' );
set( handles.buttonCalcolaIndici, 'Visible', 'off' );
```

```
function editTipoRicircolo_Callback(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.
function editTipoRicircolo_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in buttonCalcolaIndici.
function buttonCalcolaIndici_Callback(hObject, eventdata, handles)
%apro la finestra 'finestraIndici':
finestraIndici();


% --- Executes during object creation, after setting all properties.
function editNodo_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in popupTipoNodo.
function popupTipoNodo_Callback(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.
function popupTipoNodo_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function editTipoNodo_Callback(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.
function editTipoNodo_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes during object creation, after setting all properties.
function schermataPrincipale_CreateFcn(hObject, eventdata, handles)
```

```
% --- Executes on button press in buttonAnnulla.
function buttonAnnulla_Callback(hObject, eventdata, handles)

set( handles.panelInserisciRicircolo, 'Visible', 'off' );
set( handles.popupRicircolo, 'Enable', 'on' );
set( handles.panelInfoRicircolo, 'Visible', 'on' );
set( handles.buttonCalcolaIndici, 'Visible', 'on' );
value = get( handles.popupCatRicircolo, 'Value' );
string = get( handles.popupCatRicircolo, 'String' );
tipo = string{value};


% --- Executes on button press in buttonSalva.
function buttonSalva_Callback(hObject, eventdata, handles)
global infoRicircolo;

set( handles.panelInserisciRicircolo, 'Visible', 'off' );
set( handles.popupRicircolo, 'Enable', 'on' );
set( handles.panelInfoRicircolo, 'Visible', 'on' );
set( handles.buttonCalcolaIndici, 'Visible', 'on' );
%memorizzo nell'array di celle 'infoRicircolo' il tipo di ricircolo
value = get( handles.popupCatRicircolo, 'Value' );
string = get( handles.popupCatRicircolo, 'String' );
tipo = string{value};
if strcmp(tipo, '..altro..')
    tipo = get(handles.editSpecificare, 'String');
    string{value} = tipo;
    string{value+1} = '..altro..';
    set( handles.popupCatRicircolo, 'String', string );
    set( handles.editSpecificare, 'String', '' );
    set( handles.popupCatRicircolo, 'Value', 1 );
    set( handles.panelSpecificare, 'Visible', 'off' );
    %salvo nel file 'popupCatRicircolo.txt' la stringa 'stringaCat' con le
    %categorie aggiornate
    oldFolder = pwd;
    cd ..;
    percorsoFile = pwd;
    percorsoFile = strcat(percorsoFile, '\icone+popup\popupCatRicircolo.txt');
    cd(oldFolder);
    %apro il file in modalità di scrittura:
    file = fopen(percorsoFile, 'wt');
    %memorizzo nella stringa 'stringaCat' tutte le categorie che ora si
    %trovano nell'array di celle 'stringa':
    nCategorie = length(string)-1;
    stringaCat = '';
    for cont = 1:nCategorie
        stringaCat = strcat(stringaCat, string{cont}, '\n');
    end;
    stringaCat = strcat(stringaCat, '..altro..');
    %scrivo nel file il contenuto di 'stringaCat':
    fprintf(file, stringaCat);
    %chiudo il file:
    fclose(file);
```

```
end;
set( handles.textTipoRicircolo, 'String', tipo );
numRicircolo = get( handles.popupRicircolo, 'Value' ) - 2;
infoRicircolo{1, numRicircolo} = tipo;


% --- Executes on selection change in popupCatRicircolo.
function popupCatRicircolo_Callback(hObject, eventdata, handles)

value = get(handles.popupCatRicircolo, 'Value');
string = get(handles.popupCatRicircolo, 'String');
tipo = string{value};
if strcmp(tipo, '..altro..')
    set(handles.panelSpecificare, 'Visible', 'on');
else
    set(handles.panelSpecificare, 'Visible', 'off');
end;


% --- Executes during object creation, after setting all properties.
function popupCatRicircolo_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in rbPositivo.
function rbPositivo_Callback(hObject, eventdata, handles)


% --- Executes on button press in rbNegativo.
function rbNegativo_Callback(hObject, eventdata, handles)


function editAltro_Callback(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.
function editAltro_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function editSpecificare_Callback(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.
function editSpecificare_CreateFcn(hObject, eventdata, handles)
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on key press with focus on popupCatRicircolo and none of its
controls.
function popupCatRicircolo_KeyPressFcn(hObject, eventdata, handles)
```

```
function [ T, indiceT ] = spostaColonne ( T, indiceT, arrayColonneDaSpostare )
%questa funzione viene richiamata dalla funzione 'ordinaMatrici' e serve per
%spostare le colonne della matrice data in input

for cont = arrayColonneDaSpostare
    indiceT = indiceT+1;
    colonnaDaSpostare = T(:, cont);
    for cont2 = cont : -1 : indiceT+1
        T(:, cont2) = T(:, cont2-1);
    end;
    T(:, indiceT) = colonnaDaSpostare;
end;

end
```

```
function [ strCicli, percorsomax ] = trovaCicli(MI, nodoPartenza, nodo, prec)
%'trovaCicli' è una funzione ricorsiva che restituisce la stringa 'cicli'
%contenente tutti i cicli salvati uno di seguito all'altro, separati da ';'
%inoltre la funzione 'trovaCicli' restituisce la stringa 'percorsomax' con
%il percorso più lungo del nodo dato in input chiamando la funzione
global strCicli;
global percorsomax;

%se gli argomenti in input sono solo 2 significa che la funzione è stata
%chiamata dalla funzione 'arrayCicli' e non da se stessa: per cui imposto i
%valori iniziali
if nargin == 2
    nodo = nodoPartenza;
    prec = '';
    percorso = '';
end;

%aggiorno la var 'percorso' aggiungendo a 'prec' (che contiene tutti i nodi
%precedenti al nodo considerato) il nodo 'nodo' che sto considerando; prima
%però controllo di non esser già passato per 'nodo' (se l'ho già visitato
%allora la sua riga è =Nan):
if ~isnan(MI(nodo, 1))
    percorso = strcat (prec, int2str(nodo), ',');
    %se 'percorso' contiene + nodi di 'percorsomax' allora aggiorno
    %'percorsomax' ponendolo = a 'percorso';
    %'findstr(percorso, ',')' restituisce gli indici delle posizioni di
    %'percorso' contenenti ',' ;
    %'length(findstr(percorso, ','))' conta il numero di ',' in
    %'percorso', che di riflesso è = al numero di nodi di 'percorso';
    if ( length(findstr(percorso, ',')) > length(findstr(percorsomax, ',')) )
        percorsomax = percorso;
    end;
    %nella var 'successori' salvo tutti i nodi successori a 'nodo':
    successori = find( MI(nodo, :) );
else
    successori=[];
end;

%pongo =NaN tutta la riga 'nodo' della matrice di Incidenza per "tagliare"
%i legami di 'nodo' con i successori (appena salvati in 'successori'), in
%modo da evitar che il programma vada in loop quando trova un ciclo
MI(nodo, :) = NaN;

%richiamo la funzione per ogni successore di 'nodo': se un successore è =
%al 'nodoPartenza' allora ho trovato un ciclo e lo vado a memorizzare nella
%stringa 'strCicli'
for cont = successori
    trovatoCiclo = false;
    %quando trovo un ciclo, cioè quando il nodo 'cont' che sto considerando
    %è = al nodo di partenza 'nodoPartenza', aggiorno la var stringa
    %'strCicli' con un ulteriore ciclo 'percorso'; se non ho trovato un ciclo
    %allora richiamo la funzione passando il nuovo nodo 'cont'
    if cont == nodoPartenza
        trovatoCiclo = true;
```

```
        strCicli = strcat (percorso, int2str(cont), ';', strCicli);
    else
        trovaCicli (MI, nodoPartenza, cont, percorso);
    end;
end;

end
```

Bibliografia

➢ Articoli:

Allesina S., 2004. *Ecological flow networks: topological and functional features*. Thesis (PhD). University of Parma, Italy. Available from: http://www.dsa.unipr.it.

Allesina S., Azzi A., Battini D. e Regattieri A., 2010. *Performance measurement in supply chains: new network analysis and entropic indexes*. International Journal of Production Research, 48 (8), 2297–2321.

Augustinovics M., 1981. *Methods of international and intertemporal comparison of structure*. In: Carter, A.P., Brody, A. (Eds.), Contributions to Input–Output Analysis, vol. 1. North-Holland, Amsterdam.

Battini D., Allesina S. e Persona A., 2006. *Toward a use of network analysis: quantifying the complexity of supply chain network*. In: Proceedings of 4th international conference on supply chain management and information systems SCMIS 2006, 5-7 July Taichung, Taiwan, 85–902.

Blackburn J. D., Guide Jr. V. D. R., Souza G. C. e Van Wassenhove L. N., 2004. *Reverse Supply Chains for Commercial Returns*. California Management Review, 46 (2), 6-22.

DeAngelis D., Mulholland P., Palumbo A., Steinman A., Huston M., Elwood J., 1989. *Nutrient dynamics and food-web stability*. Annu. Rev. Ecol. Syst. 20, 71–95.

Finn J.T., 1976. *Measures of ecosystem structure and function derived from analysis of flows*. J. Theor. Biol. 56 (2), 363–380.

Fleischmann M., 2001. *Reverse logistics network structures and design*. ERIM Report Series Research In Management, 52.

Francas D. e Minner S., 2009. *Manufacturing network configuration in supply chains with product recovery*. Omega 37 (4), 757-769.

Giuntini R. e Gaudette K., 2003. *Remanufacturing: The next great opportunity for boosting US productivity.* Business Horizons, 46 (6), 41−48.

Guide Jr. V. D. R. e Van Wassenhove L. N., 2002. *The Reverse Supply Chain*. Harvard Business Review, 80 (2), 25-26.

Guide Jr. V. D. R. e Van Wassenhove L. N., 2009. *The Evolution of Closed-Loop Supply Chain Research*. Operations Research 57 (1), 10-18.

Guo P., Song J. S. e Wang Y., 2010. *Outsourcing structures and information flow in a three-tier supply chain*. International Journal Of Production Economics, 128 (1), 175-187.

Hannon B., 1973. *The structure of ecosystems*. J. Theor. Biol. 41, 535–546.

Herendeen R., 1989. *Energy intensity, residence time, energy, and ascendancy in dynamic ecosystems*. Ecol. Modelling 48 (1–2), 19–44.

Higashi M., Patten B. C., Burns T. P., 1991. *Network trophic dynamics: an emerging paradigm in ecosystems ecology*. In: Higashi, M., Burns, T.P. (Eds.), Theoretical Studies of Ecosystems-The Network Perspective. Cambridge University Press, Cambridge.

Hutchinson G. E., 1948. *Circular causal systems in ecology*. Ann. N. Y. Acad. Sci. 50, 221–246.

Kannan G., Sasikumar P. e Devika K., 2010. *A genetic algorithm approach for solving a closed loop supply chain model - A case of battery recycling*. Applied Mathematical Modelling, 34 (3), 655-670.

Kara S., Rugrungruang F. e Kaebernick H., 2007. *Simulation modelling of reverses logistics networks*. Int. J. Production Economics, 106 (1), 61-69.

Kusumastuti R. D., Piplani R. e Lim G., 2004. *An approach to design reverse logistics networks for product recovery*. In Proceedings of IEEE international engineering management conference, 1239-1243.

Lamsali Hendrik e Liu Jiyin, 2008. *Optimizing the selection of product recovery options*. IEEE International Conference on Industrial Engineering and Engineering Management, 1704-1708.

Leontief W., 1963. *The Structure of American Economy 1919–1939*. Oxford University Press, New York.

Loreau M., 1994. *Material cycling and the stability of ecosystems*. Am. Nat. 143, 508–513.

Pagell M., Wu Z. e Murthy N. N., 2007. *The supply chain implications of recycling*. Business Horizon, 50 (2), 133-143.

Ruggeri Laderchi D. e Payaro A., 2004. *Operatori e ruoli per una gestione efficiente della logistica inversa*. Pubblicato su: Logistica. Disponibile presso il sito: http://www.reloaderitalia.it/doc/download/gest_effic_logistica_inversa.pdf

Salema M. I. G., Barbosa-Povoa A. P. e Novais A. Q., 2007. *An optimization model for the design of a capacitated multi-product reverse logistics network with uncertainty*. European Journal of Operational Research, 179 (3), 1063-1077.

Shannon C. E. e Weaver W, 1948. *The mathematical theory of communication*. Urbana, IL: University of Illinois Press.

Simpson Dayna, 2010. *Use of supply relationships to recycle secondary materials*. International Journal of Production Research, 48 (1), 227-249.

Thierry Martijn, Salomon Marc, Van Nunen Jo e Van Wassenhove L. N., 1995. *Strategic Issues in Product Recovery Management*. California Management Review, 37 (2), 114-135.

Ulanowicz R.E. e Kay J., 1991. *A package for the analysis of ecosystem flow network*. Environmental Software, 6 (3), 131-142.

Van Nunen Jo e Zuidwijk Rob, 2004. *E-Enabled Closed-Loop Supply Chains*. California Management Review, 46 (2), 40-54.

Wang Zheng e Bai Hua, 2010. *Reverse Logistics Network: A Review.* IEEE International Conference on Industrial Engineering and Engineering Management, 1139-1143.

➢ Libri:

Blumberg Donald F., 2005. *Introduction to Management of Reverse Logistics and Closed Loop Supply Chain Processes*. CRC Press edition.

Dekker R., Fleischmann M., Inderfurth K. e Van Wassenhove L. N., 2004. *Reverse Logistics: Quantitative Models for Closed-Loop Supply Chains*. Springer, Hardbound.

➢ Siti:

Apple Store (http://store.apple.com/it)
26 agosto 2011

Ecodom (http://www.ecodom.it/raee/la_normativa.aspx)
28 settembre 2011

Pubblica Amministrazione
(http://www.pubblicaamministrazione.net/governance/news/2782/raee-il-progetto-scuolambiente.html)
29 settembre 2011

Ecolight (http://ecolight.nemo.it/attachp/Brochure%20Smaltimento%20Ecolight.pdf)
1 ottobre 2011

Ringraziamenti

Ringrazio la Professoressa Daria Battini e l'Ingegnere Anna Azzi per la cura con cui mi hanno seguito durante tutto il lavoro di tesi.

Particolare riconoscenza va alla morosa Chiara che (finora) ha sopportato tanti discorsi noiosi da ing. e che (finora) è sopravvissuta a herpes, allergie e virus di tutti i tipi...

Un pensiero affettuoso va alla ormai storica compagnia e ad amici/amiche fidati/fidate che ci sono sempre per stratracannare e stramaledire le donne, il tempo ed il governo (tratto da De Andrè, 1966 :-D).

Grazie a tutti i parenti, zii, zie, cugini, cugine di mare, città e... campagna!

Ma il ringraziamento più grande è rivolto ai miei genitori che mi hanno sostenuto (anche finanziariamente!!!) lungo tutto il percorso di studi: senza i loro insegnamenti non sarei arrivato a questo importante traguardo.

GRAZIEEEEEEEEEE a tutti…