Università degli Studi di Salerno

Facoltà di Scienze Matematiche Fisiche e Naturali
Dipartimento di Matematica

Dottorato di Ricerca in Matematica
XII Ciclo – Nuova Serie

TESI DI DOTTORATO

Analisi e risoluzione del problema dell'allocazione ottima delle risorse

CANDIDATO: **DOTT. ANTONIO GIUSEPPE DE PASCALE**

COORDINATORE: **PROF.SSA PATRIZIA LONGOBARDI**

TUTOR: **PROF. CIRO D'APICE**

Anno Accademico 2013 – 2014

Indice

Introduzione

Il problema di assegnare un determinato compito ad un lavoratore fa parte del più ampio settore delle problematiche inerenti la gestione delle risorse umane.
Per gran parte dei due decenni appena trascorsi, ha largamente dominato l'idea secondo la quale l'innovazione di tipo tecnologico fosse la principale fonte del successo delle imprese; mentre l'organizzazione e le modalità di gestione delle risorse umane hanno ricevuto una attenzione secondaria. Si è scoperto poi che una maggiore crescita produttiva scaturisce quasi sempre dalla stretta complementarità tra investimenti in nuove tecnologie e investimenti in struttura organizzativa; tale aumento di produttività è stato tradotto poi in crescita della *performance*, che è l'obiettivo del nostro approccio a queste tematiche.
La conoscenza generata dall'innovazione tecnologica, per essere accolta, necessita di un piano di pratiche organizzative adeguate. Per questo motivo è sempre più stretto il nesso tra il processo di sviluppo tecnologico e innovazione nel settore della gestione e dell'organizzazione delle risorse. Il coordinamento e la gestione delle risorse umane e dei *feedback* relativi ai diversi aspetti delle specifiche attività diventa una linea d'azione strategica per le imprese al fine di ottenere performance eccelse dai propri dipendenti nonché risultati economici superiori.
Due precursori dell'introduzione dell'ICT (Information and Communication Technologies) nella gestione delle risorse umane (HR) furono Milgrom e Roberts (1990). Essi sostenevano che molte delle più importanti attività che incidono sul profilo organizzativo delle imprese (si pensi alle attività di comunicazione, monitoraggio, immagazzinamento dati, supervisione etc..) possono essere svolte con un minore impiego di risorse e una maggiore compressione dei livelli gerarchici, a patto che si rendano disponibili sistemi che consentano una riduzione dei costi dell'informazione e un incremento dei suoi flussi. Le ICT rispondono pienamente a tale esigenza, consentendo nuove strutture organizzative più orizzontali, più snelle e più atte a veicolare informazioni. Secondo questa visione le ICT, completando o sostituendo vecchie mansioni e/o creandone di nuove, influenzano radicalmente la gestione dell'assetto organizzativo-gestionale, incidendo profondamente sulle produttività relative dei lavoratori contraddistinti da diversi livelli di capitale umano.
Ci sono numerosi studi (Teece, 1996), (Pisano, 1998) che legano in maniera complementare l'evoluzione tecnologica a quella dell'Human Resource Management fornendo vantaggi alla produttività. Un ulteriore approccio teorico fu quello di considerare la produttività in maniera modulare ma costituita da elementi eterogenei e complementari (Cristini A., 2008).
Ad ogni modo, per rivelarsi pienamente profittevoli, tali mutamenti tecnologico-organizzativi necessitano di lavoratori più qualificati, più atti al *problem solving* ed in grado di svolgere allo stesso tempo molteplici compiti. In questo senso la complementarità, tra lavoratori e compiti, sarebbe un effetto del mutamento organizzativo-gestionale indotto dalle nuove tecnologie ICT.
I dati relativi ai task, le relative meta-informazioni insite negli skill dei dipendenti nonché nelle relazioni che intercorrono con le attività che svolgono (o che hanno svolto), sono una miniera di elementi utili al fine di ottimizzare l'assegnazione dei futuri lavori alle risorse umane, ottenendo performance migliorative per la risorsa e quindi per l'azienda.
Assodato che la crescita e la sostenibilità dei profitti, nonché le capacità di competere delle imprese siano strettamente legate ai modi in cui sono ripartiti i compiti e alla gestione delle risorse umane, bisogna focalizzarsi sugli strumenti ausiliari a tale trasformazione.
In sintesi, una sapiente gestione delle risorse umane e la conseguente capacità di gestire le sinergie tra innovazione tecnologica ed organizzazione consente alle imprese maggiore produttività (Breshnan, 2002), (Hunthausen, 2003), (Leoni R., 2003), (Mazzanti, 2004), (Zwick, 2004), (Cristini A., 2008), e quindi maggior redditività (Colombo, 2005), ossia incrementi salariali (Osterman, 2006). Non per ultimo, la gestione coerente delle risorse implica la crescita delle competenze dei lavoratori stessi (Leoni R., 2010). Infatti una corretta assegnazione dei task ad esatte competenze e la giusta comunicazione tra le parti, spesso si tramuta anche in un involontario processo di

apprendimento continuo che sovente garantisce l'accumulo di nuove conoscenze, principale fattore competitivo (Frasca P., 2007). Freeman, Kleiner e Ostroff (2000) si soffermarono invece sul concetto morale secondo cui la giusta assegnazione delle risorse umane porta a maggior soddisfazione e gratificazione da parte degli stessi dipendenti.
Per rivelarsi tale, questa riorganizzazione richiede l'implementazione di nuove *work practice* in larga parte incentrate su capacità di ordine cognitivo e interattivo non facilmente riproducibili da un comune e semplice elaboratore unitario o implementabili da un lavoratore scarsamente qualificato. In particolare, tali nuove modalità lavorative si esplicitano in una serie di pratiche che hanno nell'*empowerment* delle risorse umane l'elemento centrale, e che si concretizzano nella riduzione dei livelli gerarchici, nel coinvolgimento dei lavoratori, nel lavoro in team, nella pluricompetenza, nei sistemi di valutazione della performance, nelle buone relazioni tra gli attori principali e soprattutto nella "giusta" assegnazione dei compiti e delle responsabilità!
In tale ottica, acquisisce un ruolo strategico la disponibilità di un'adeguata dotazione di infrastrutture, la diffusione di modelli efficienti, ma soprattutto l'innovazione e la condivisione di informazioni e conoscenze.
Tenendo ben presente le teorie e i modelli prodotti nell'ambito della ricerca nel settore delle risorse umane abbiamo tentato di migliorare il processo di gestione dei compiti e del loro assegnamento alle risorse umane.
Nei processi di innovazione dell'HRM, in genere, intervengono almeno tre fattori rilevanti: la conoscenza posseduta, il capitale umano e l'associazione tra innovazioni organizzative e cambiamenti tecnologici. Sulla base di tale orientamento l'innovazione è, dunque, il risultato di processi di apprendimento, produzione e combinazione di conoscenza con tecnologia (Foray, 2006). Cercheremo, in questo lavoro, di applicare tali concetti alla gestione dei task relativi a malfunzionamenti informatici.

1. Motivazioni

Taylor, nello *Scientific Management* (1911), asseriva che la disciplina nel lavoro, orientata all'organizzazione meticolosa delle mansioni e del tempo di lavoro, è ritenuta l'elemento preponderante. Questo concetto, che presuppone alla base un'efficiente organizzazione del lavoro, non poteva essere concepito e controllato esclusivamente dal singolo imprenditore. Per tale motivo nacque la cosiddetta "burocrazia aziendale", strutturata in modo gerarchico, che si distribuiva equamente compiti e responsabilità, lasciando all'imprenditore esclusivamente decisioni di controllo e indirizzo complessivo: stava nascendo l'idea di HRM - Human Resources Management!
Appurato che i collaboratori costituiscono un vero e proprio capitale per l'organizzazione, la funzione delle *Risorse Umane* ha iniziato così ad occupare un ruolo sempre più centrale. Il mondo delle organizzazioni in questi ultimi anni sta rivolgendo sempre più l'attenzione alle risorse umane, poiché la qualità e le competenze di queste ultime costituiscono un reale vantaggio competitivo per le aziende operanti in tutti i settori. Le organizzazioni che sapranno sfruttare la creatività e l'energia dei loro collaboratori potranno fornire i prodotti e i servizi richiesti dai nuovi mercati.
Si sta dunque espandendo la convinzione che una corretta gestione delle risorse umane possa aiutare alla determinazione del successo del proprio business, e possa costituire un elemento fondamentale da tenere in considerazione per la formulazione della strategia dell'organizzazione.
In questa sede approfondiremo le relazioni sul terreno della gestione delle risorse umane per ottenere la massima produzione nel più breve tempo possibile e al minor costo, attraverso una buona combinazione tra sapiente utilizzo delle abilità richieste alle differenti risorse che cooperano ad un progetto e ottimizzazione delle competenze possedute.
La gestione delle variabili legate all'HRM presuppone una chiara comprensione riguardo a come operare sulle risorse umane al fine di favorire l'attuazione degli obiettivi strategici di impresa, ma allo stesso tempo consente di individuare indicatori, mediante i quali sia possibile effettuare le misurazioni sulle variabili inerenti le persone stesse. Ne consegue una gestione strategica delle risorse umane che consente di mettere a punto dei procedimenti per rinnovare valori e competenze con lo scopo di ottimizzare la performance economica.
Per ottenere quindi una gestione efficiente in questo contesto, le organizzazioni devono saper sfruttare al massimo le potenzialità dei propri dipendenti, dato che l'apporto e la produttività che il capitale umano è in grado di dare all'azienda costituiscono il perno attorno al quale ruota il successo dell'impresa.
Per conseguire tale obiettivo, risulta di fondamentale importanza l'utilizzo di soluzioni tecnologicamente avanzate e innovative. Ciò implica che le funzionalità inerenti le HR devono subire un processo di trasformazione, al fine di delineare un programma completo e integrato di gestione delle risorse umane, ovvero integrare informazioni e pratiche relative ai collaboratori (carriere, collaborazioni, informazioni personali) con processi e strategie aziendali (progetti, obiettivi, mission). Per consentire e agevolare tali operazioni, le aziende dovranno dotarsi di efficaci tool mirati all'implementazione di programmi dedicati alla gestione delle risorse umane e allo stesso tempo integrati con i processi di business dell'impresa, in modo da incrementare la produttività del proprio personale. Senza l'utilizzo di tali strumenti risulterebbe alquanto difficile essere o rimanere competitivi nell'attuale sistema economico, caratterizzato da costante evoluzione e continui cambiamenti.
L'obiettivo della tesi è quello di introdurre un modello risolutivo in una piccola realtà dipartimentale, ponendo l'attenzione ai benefici che l'utilizzo di questo specifico strumento può apportare anche in realtà più grandi.
Veniamo alla gestione delle competenze che poi vedremo essere riunite sotto forma di *skill*.

All'interno dell'azienda la conoscenza subisce un processo dinamico in cui viene "*stored, processed and understood*" (Howells, 2002). In tale ottica è necessaria una gestione della conoscenza come risposta pratica al mantenimento, allo sviluppo ed allo sfruttamento delle potenzialità cognitive dell'impresa. Infatti, la principale criticità non è nel momento del reperimento della conoscenza, ma nella sua corretta interpretazione ed amministrazione. L'azienda, se è capace di gestire al meglio il processo di *formation*, può superare le limitazioni sul fronte della conoscenza. Gestire adeguatamente il capitale umano significa renderlo spendibile, utilizzabile in maniera profittevole da parte dell'impresa. Quando tali processi funzionano correttamente, i lavoratori dispongono di maggiori conoscenze acquisite o sviluppate *on the job* e contribuiscono in maniera più efficace all'incremento di produttività.
Il modelli studiati per la gestione e l'assegnazione delle competenze, di cui sarà fatto ampio cenno nel Cap. 3, prevedono spesso l'utilizzo di una rete ad alto impatto tecnologico. Già ad inizio del nuovo millennio d'altronde la struttura tecnologica, detta *K-technology*, veniva individuata come requisito fondamentale per la gestione della conoscenza all'interno delle organizzazioni (Gravili, 2000) e (Huysman, 2002). Tali tecnologie, come le intendiamo noi nel nostro studio, non hanno senso se considerate singolarmente. Esse avranno lo scopo di favorire la comunicazione tra soggetti diversi (pubblica amministrazione/consumatori/imprese), che hanno la possibilità di attingere alle informazioni ed assorbire nuove conoscenze da una sorta di memoria collettiva. In tale ottica, esse saranno oggetto del più svariato impiego, sempre più spesso utilizzate per accrescere la qualità del lavoro, migliorare le condizioni di vita, fornire opportunità di apprendimento, migliorare la produzione e la trasformazione dei prodotti/servizi, eliminare le barriere tra i mercati, etc.. Inoltre l'ICT, attraverso le molteplici applicazioni a cui si presta grazie alle sue doti di elevata adattabilità, semplifica e velocizza i compiti da svolgere nei diversi comparti di un'organizzazione, promuovendo l'aumento della produttività. Quest'ultimo si realizza anche grazie alla maggiore disponibilità di informazione in tempo reale per elaborare soluzioni migliori a complessi problemi di ottimizzazione.
Consideriamo pertanto il problema della formazione di team a partire da una comunità di persone interconnesse che posseggano diversi insiemi di profili (skill). Ogni task richiede un insieme di skill che può essere coperto dai membri del team assemblato. Siamo interessati in particolare al problema dell'allocazione ottima dei membri di un'azienda su di un lavoro, tenendo conto delle reti di relazioni con gli altri agenti. Considereremo un modello dove ci sono tipi multipli di task, e ogni membro ha una certa abilità cognitiva per ogni tipo di compito. Ogni membro è perciò caratterizzato dalle sue abilità nel realizzare ognuna delle differenti tipologie di task. L'azienda, in genere, ha un insieme di progetti da eseguire e ogni progetto consiste di un insieme di task di differenti tipi. Conseguentemente, ogni task è caratterizzato dal valore, dalla tipologia, e dal progetto di cui fa parte. L'idea motivazionale è che l'organizzazione è in realtà un gruppo di esperti. Ogni esperto ha esperienza in un particolare tipo di task a prescindere dal progetto a cui questo task appartiene. Gli esperti di alcuni tipi di task, se sono oggetto di una condivisione fruttuosa dell'informazione, possono avere effetti sinergistici sulle performance di altri diversi task dello stesso progetto.
Una forte spinta motivazionale, che ha orientato il sottoscritto ad indirizzare i propri studi dottorali verso la problematica dell'assegnamento delle risorse umane a task, è data dalle peculiarità del lavoro che quotidianamente svolgo.
Sono un informatico e, attualmente, presto il mio servizio per il Ministero dell'Interno. In particolare sono il responsabile dei Sistemi Informativi di un piccolo dipartimento afferente agli Affari Interni e Territoriali che si occupa di gestire le carriere dei Segretari Comunali e Provinciali e segue alcuni aspetti della Scuola Superiore della Pubblica Amministrazione Locale.
Oltre alle pianificazioni, progettazioni e manutenzioni informatiche ordinarie, gran parte del mio lavoro è indirizzato al problem solving.
La risorsa principale che ho a disposizione è il personale (con le sue competenze) che deve essere gestito in modo appropriato per far raggiungere al servizio fornito alti livelli di efficacia ed efficienza

anche in vista di evoluzioni dell'attività inquadrata e continuamente modellata da Decreti Ministeriali.
La pianificazione dei turni e dei carichi attualmente è svolta a mano dal sottoscritto che cerca di ripartire equamente sui collaboratori il carico di lavoro tenendo in considerazione le disponibilità e le competenze.
Questo modo di gestire il personale, oltre a richiedere molto tempo per la fase decisionale, non garantisce la formazione di team di lavoro omogenei per esperienza e capacità a causa delle numerose variabili e dei molti vincoli da soddisfare, generando talvolta insoddisfazione.
Per risolvere in modo ottimale questo problema ho pensato di impiegare tecniche di ricerca operativa e di valutare i risultati ottenuti dall'elaborazione su calcolatore.
Colleghi e utenti del mio ufficio spesso si trovano davanti a problematiche di tipo informatico o simili che non sono in grado di superare e che, se non risolte nei giusti tempi, potrebbero arrecare danni anche seri all'economia dell'Ente. In una sempre più comune ottica di razionalizzazione della spesa inoltre c'è regolarmente necessità di diminuire i "tempi morti" che rappresentano un altro costo aggiuntivo per il mio Ufficio.
Una scarsa efficienza nell'utilizzo del personale è causa di costi diretti e indiretti di gestione molto elevati e di frequenti ritardi nelle consegne, che compromettono a volte le performance dell'Ente.
L'obiettivo primario del mio lavoro è quindi quello di migliorare la risoluzione dei guasti informatici con una più efficiente e gratificante gestione del team di lavoro che dirigo.
Gli studi sull'ottimizzazione del processo di assegnazione di risorse umane a lavori sono ancora pochi. Tralasciando gli aspetti formativi, un ambito ancora poco studiato riguarda la giusta costituzione dei team di lavoro e la loro incidenza sulla produttività, che è proprio l'argomento specifico che ci siamo proposti di indagare.
Preliminarmente abbiamo riordinato gli studi circa il problema originario, dandone le giuste motivazioni ed inquadrandone il contesto.
Nella seconda parte della tesi invece abbiamo studiato una corretta risoluzione della problematica dandone una modellazione scientifica, inquadrata nell'abito della Ricerca Operativa.
Più in dettaglio il lavoro è strutturato come segue. Nel primo capitolo viene affrontato il problema dell'assegnazione in maniera descrittiva con riferimento alle teorie scientifiche, filosofiche ed economiche preesistenti. Nel secondo capitolo sono menzionati gli approcci usuali alla risoluzione del problema dell'HRM, con particolare riferimento alla formazione di team. Nel terzo capitolo è descritto lo stato dell'arte circa le metodologie scientifiche del team building facendo riferimento ai diversi approcci matematici. Nella quarta parte vi è la descrizione dello scenario attuale, in cui troverà applicazione la nuova metodologia. Nel quinto capitolo è descritto il nuovo approccio risolutivo. Nell'ultimo capitolo, dopo alcune sperimentazioni, sono valutati gli aspetti positivi introdotti dalla nuova metodologia. In fine: conclusioni e sviluppi futuri.

2. La selezione di un Team

Prima di fare un'ampia panoramica sullo stato dell'arte circa i modelli di ausilio alla selezione e formazione di un team di esperti, è necessario introdurre i concetti che sono alla base delle logiche che governano i criteri per la selezione di una squadra di lavoro. Inoltre è utile anche accennare ad alcuni aspetti collaterali che servono ad arricchire la formulazione dei modelli risolutivi.
Dapprima definiamo cos'è il *Team Building*: è una vera e propria filosofia di job design nella quale il dipendente è visto come membro di team interdipendenti invece che come lavoratore individuale. Nell'ambito delle risorse umane, esso costituisce un insieme di attività selettive e formative, il cui scopo è la formazione di un gruppo di persone. L'obiettivo di tale filosofia è quello di formare team "giusti" per raggiungere l'obiettivo nei tempi prefissati, nonché incrementare dinamicamente le competenze dei vari team che cooperano.
All'interno di questo insieme di metodologie abbiamo distinto il *Team Selection* dal *Team Formation.* Entrambi gli scenari hanno in comune una collezione di task da svolgere ed un insieme di utenti-lavoratori. Dato che gli utenti hanno differenti livelli di esperienza, l'obiettivo del problema di selezione del team è quello di trovare una allocazione dei task agli utenti tale da massimizzare il livello di competenza totale. Nel team formation invece si punta ad ottimizzare parametri differenti e vincoli reali (capacità, collaborazioni passate, prossimità,…), come vedremo approfonditamente nel successivo capitolo. Nel team selection inoltre non è definita alcuna rete sociale tra gli utenti e il problema è osservato solamente dal punto di vista dell'allocazione di risorse (molto meno pratico).
Per rendere quanto più vicino al mondo reale il mio studio ho deciso di introdurre brevemente il concetto di multi-progetto: in azienda, nell'ottica di ridurre i costi e massimizzare l'utilizzo delle competenze, più lavoratori collaborano a diversi progetti in simultanea. In ambienti di questo tipo ovviamente ha senso parlare di Team formation!
Visto che molte strategie si basano sui giudizi che un determinato lavoratore raccoglie nella corso della sua carriera e soprattutto sul grado di fiducia che egli riscuote nei confronti dei suoi colleghi o superiori, sembrava opportuno contestualizzare l'idea di feedback nel mondo del team building.
Alla fine di questo capitolo saranno introdotte le nozioni comuni a tutti i modelli, con particolare attenzione ai vincoli considerati, alle performance attese ed ai costi sotto diversi punti di vista.

2.1 HRM e i multi-progetto

Nelle realtà medio piccole, come quella in cui mi adopero, ci si trova spessissimo a dover gestire l'esecuzione di più progetti contemporaneamente. I progetti vengono raggruppati sia per facilitarne la conduzione che per migliorare la gestione di chi lavora ad essi. Inoltre, così facendo si cerca di raggiungere l'obiettivo strategico di ogni progetto, e indirettamente si massimizza quello dipartimentale.
Nella maggioranza dei casi, i progetti richiedono l'utilizzo di differenti tipologie di risorse, ad esempio nel mio ambito: sistemista, amministratore di rete, sviluppatore junior, sviluppatore senior…
Tali progetti, inoltre, possono avere differenti gradi di difficoltà e rischi, ad esempio:

1. Semplici review o repliche di qualcosa già implementato in passato
2. Nuovi sviluppi su tecnologie comunque conosciute
3. Nuove realizzazioni su tecnologie inedite o sulle quali non esiste un know-how specifico

Per queste categorie potrebbe essere sintetizzata la seguente semplice tabella che si rifà alla più generica classificazione e assegnazione dei rischi e delle complessità relativi alle varie attività.

Tipo Progetto	Rischi	Difficoltà
Review o repliche	Bassi/Inesistenti	Nulla
Nuovi sviluppi	Bassi	Minima
Nuovi sviluppi su tecnologie inedite	Alti	Alta

Tabella 1: Assegnazione di rischi a difficoltà

Le risorse a più alta specializzazione o responsabilità sono quelle a maggior costo e spesso anche quelle meno numerose o disponibili. Si pone quindi la chiara necessità di gestire al meglio tali progetti cercando il miglior bilanciamento economico, di risorse e di gestione del progetto stesso. In altre parole la loro gestione potrebbe correre su linee parallele.
Gestire progetti tra loro disgiunti presenta sicuramente una maggiore complessità, però ha il suo lato positivo, ossia il *bilanciamento*.
Potremo bilanciare le risorse tra progetti, rallentando o velocizzando lo sviluppo dell'uno o dell'altro a seconda delle necessità e dei vincoli di ognuno. Così facendo, bilanceremo anche i rischi legati alla corretta esecuzione dei progetti.
La gestione multi-progetto non è un'attività semplice. La creazione di un processo strutturato, in continua evoluzione, rifinitura e controllo, rappresenta forse l'unico vero strumento per far fronte a complessità e rischi collegati.
Per poter portare avanti un discorso del genere è imprescindibile approcciare alle teorie del team building: bisognerà seguire rigorosi approcci scientifici alla formazione dei team di lavoro.
Spesso e volentieri il concetto di progetto e quello di team si scambiano vicendevolmente il significato perché al progetto corrisponde il team con un rapporto di n:n .

2.2 I feedback e l'agilità nel Team Building

La logica dell'organizzazione come strumento e quella della comunicazione come semplice servizio per l'informazione è stata superata nelle nuove pratiche lavorative, che concepiscono il management delle informazioni come uno strumento capace di mettere in comune, dare e ricevere un feedback, scambiare dati lavorativi.
Ad oggi le imprese che ottengono i migliori risultati economici e produttivi riescono a combinare forti investimenti in nuove tecnologie con l'impiego di pratiche innovative nell'organizzazione del lavoro, come ad esempio i gruppi di lavoro e i gruppi di problem solving. Il concetto di feedback si evolve di pari passo con quello di team. Per migliorare le performance di un team, ma anche solo definire lo stato delle forze attuali e le relative debolezze possono essere utilizzati i feedback all'interno dei team. Inoltre sarà possibile anche verificare quanto dista lo stato dei lavori attuale da quello realmente desiderato: questa è la base della *gap-closure strategy*.
L'importanza del gruppo è risaputa e assodata. E' vero, il singolo è agile e risponde velocemente ai cambi di scenario, ma la fragilità dovuta a risorse limitate gli precludono molte delle strade percorribili. La forza del gruppo deriva dal fatto che questo, nel suo complesso, ha risorse e conoscenze maggiori e, in teoria, nessun obiettivo gli è precluso. La minore agilità potrebbe però pesare in situazioni che richiedono forte dinamicità.
La *teoria dell'agilità* ci insegna che se un team, anche se contenuto nei numeri, ha le caratteristiche qui sotto elencate, riesce ugualmente a fare fronte a situazioni di forti cambiamenti e flessibilità.

Meglio è strutturato un team e più i propri membri acquisiranno consapevolezza dei mezzi posseduti e dei gap da colmare.
Bisogna saper individuare le caratteristiche adatte al nostro team:

- Competenze tecniche professionali elevate dei singoli
- Dimensione ridotta del team
- Personalità dei membri orientata alla collaborazione
- Buone capacità di adattamento
- Disponibilità e apertura dei singoli all'apprendimento

Un buon tool quindi dovrà possedere la capacità di saper selezionare anche in base agli elementi succitati.
Campanella (2005) osserva che per formare un buon team è necessario che i membri percepiscano un senso di coesione ed interdipendenza, prendendo coscienza delle proprie diversità professionali e della necessità degli altri per raggiungere gli obiettivi. L'unione in funzione di un risultato e la consapevolezza dell'interdipendenza tra le funzioni sono due aspetti fondamentali per creare un clima di fiducia e collaborazione reciproca (*trust*).
I gruppi di lavoro sono comunemente considerati come la pratica che caratterizza maggiormente i nuovi modelli produttivi-organizzativi. In questa situazione l'individuo conserva tutto il suo valore, ma allo stesso tempo si integra con gli altri.
Il **team building** in generale indica specifiche metodologie nate e sviluppate per lavorare sui gruppi ed in particolare su team di lavoro task-oriented. La sua attività si focalizza nello sviluppo delle competenze distintive di un'azienda andando a creare un senso di identità su ogni componente del gruppo di lavoro. La costruzione del gruppo può avere una valenza formativa, se associata ad un'analisi dettagliata dei bisogni, inoltre tale attività stimola le aziende a riflettere sull'importanza di lavorare in contesti relazionali piacevoli. L'obiettivo di tale filosofia è anche diversificare il concetto di responsabilità che da individuale diviene di gruppo con riflessi nella produttività collettiva.
I vantaggi di tale approccio sono: maggiori benefici nell'ottimizzazione dei tempi durante i cambiamenti organizzazionali; aumento delle comunicazioni e quindi delle collaborazioni e maggiore inclinazione alla flessibilità di pensiero; duttilità degli skill e delle competenze.
Per quanto in seguito, nell'elaborazione di questo lavoro, si seguiranno rigidi criteri scientifici, la costruzione di un team di lavoro non può prescindere dalle seguenti caratteristiche: le esperienze degli individui candidati (abilità nel problem solving), relationship (capacità di rilasciare e ricevere feedback), chiarezza ed onestà nel dichiarare l'obiettivo comune, competenze del leadership (orientate all'obiettivo finale), capacità di coinvolgimento e comunicatività. Alcune di queste caratteristiche (soft skill), dopo la loro formalizzazione, saranno utili a costruire i profili utenti, come vedremo nel cap. 5.2.

2.3 Team Formation vs Team Selection

Il problema della selezione di un team, studiato moltissimo nella comunità di Ricerca Operativa, è una delle chiavi di successo di ogni organizzazione. Le aziende più affermate, generalmente, prediligono la formazione di team per portare a termine task complicati. All'interno del processo di formazione delle squadre di lavoro, vi è innanzitutto quello della selezione (seguono quello di analisi, confronto, learning, scelta dei parametri da potenziare…).
Non bisogna infatti confondere la selezione del team con il team formation. Entrambi gli scenari hanno in comune un insieme di lavoratori ed una serie di task da svolgere. Dato che gli utenti hanno differenti livelli di esperienza, l'obiettivo del team selection è quello di trovare una allocazione dei compiti ai lavoratori tale da massimizzare la competenza complessiva. Nel team formation invece si punta ad ottimizzare, oltre che le competenze anche altri parametri, come vedremo

approfonditamente in seguito. Mentre nel team formation il tutto è contestualizzato su di un social network tra gli user, nel team selection il problema è osservato solamente dal punto di vista dell'allocazione di risorse e quindi senza considerare le relazioni tra gli utenti (per cui meno reale).
Nel corso degli anni dopo che molti studi avevano messo in evidenza che la performance di un individuo in un gruppo dipende non solo dai suoi skill, ma anche dalle sue relazioni con gli altri membri del gruppo, molti ricercatori si sono orientati allo specifico processo del *team formation*, tralasciando quello più generico della *selection*. Le relazioni tra i membri del team non necessariamente devono essere di amicizia, è importante che i membri lavorino insieme sinergicamente e che lavorino in ambienti collaborativi. Le organizzazioni stanno cercando sempre più di ottimizzare la formazione del team, tenendo conto ad esempio delle diverse personalità dei lavoratori da assegnare.
Il concetto di team formation, o joint action, è centrale in diverse discipline (teoria computazionale dell'organizzazione, sistemi di e-learning, intelligenza artificiale distribuita, …). L'uso di un social network fornisce vantaggi anche in merito al fatto che i team che hanno lavorato insieme precedentemente, comportano costi di comunicazione più bassi, e quindi un risultato migliore.
La cooperazione, la collaborazione, la delega, l'allocazione di risorse, la distribuzione di skill e altri fattori sono influenti sull'abilità dell'organizzazione ed incoraggiano le azioni collettive degli individui. Bisogna per cui comprendere l'impatto del design e della struttura della rete sulle dinamiche dell'azienda stessa.
Nel corso degli anni, infatti, sono emersi due trend interdisciplinari, che hanno avuto un grande impatto sullo studio delle organizzazioni: il modello computazionale agent-based e lo studio della struttura nonché delle dinamiche di reti complesse (per dettagli Cap.3).
Tali filoni di studio hanno impattato su molte discipline quali fisica, chimica, biologia, ecologia, scienze sociali, medicina e informatica (Carley, 1999). Gli studi sul team formation e la cooperazione nelle dinamiche organizzazionali hanno suggerito l'importanza della struttura a rete sulla performance dell'azienda. *Huberman* e *Hogg* (1991) già teorizzavano un approccio probabilistico allo studio organizzazionale per trasformare le organizzazioni in larga scala, trovando che le unioni di imprese (cioè quelle in cui ogni agente ha un piccolo numero di vicini) sono più stabili rispetto alle aziende connesse, seppur in maniera stretta. Questi studi non si focalizzano sulle performance del task, ma evidenziano l'importanza delle varie caratteristiche delle network organizzazionali. Similarmente, *Glance* e *Huberman* (1991) mostrano che la cooperazione è più probabile nelle organizzazioni che sono gerarchicamente strutturate con raggruppamenti "fluidi", ossia variabili. *Miller* (2001) dimostra l'importanza delle strutture a rete sui processi di informazione e suggerisce meccanismi per far evolvere un'organizzazione attraverso performance più efficienti. Per studiare l'adozione e lo spread delle convenzioni sociali nell'organizzazione, *Delgado* (2002) valuta l'efficienza organizzazionale di questa dinamica sulle reti complesse.

2.4 La comunicazione: sia mezzo e che struttura sociale

La comunicazione tra le parti è un elemento di integrazione, di scambio e di coesione, capace di fluidificare e di collegare tutto il sistema aziendale e inter-aziendale. Qualità ed efficienza dell'informazione, apertura e trasparenza nei rapporti sono i fattori alla base di un positivo clima di cooperazione.
Come già accennato, dopo alcuni studi sui vecchi modelli di gestione HR, rapportati ai nuovi concetti legati all'ICT, si è dedotto che una struttura a rete, potrebbe rendere meglio gestibili le competenze nell'ambito delle risorse umane disponibili.
In un contesto caratterizzato da una competizione più intensa, da costi crescenti, dalla rapida obsolescenza delle tecnologie, le aziende o enti hanno bisogno di aggregare dipartimenti per risparmiare tempo e risorse, riducendo drasticamente la complessità dell'organizzazione. All'aggregazione delle organizzazioni, consegue anche la somma dei loro skill e quindi delle risorse,

dando vita ad una struttura *virtuale* o *dinamica*. Il termine rete, o *network*, è utilizzato per identificare un'ampia gamma di relazioni sia interorganizzative che interpersonali, diverse non solo per la natura delle attività nelle quali si concretizza la cooperazione, ma anche per le aree organizzative coinvolte, l'intensità dei rapporti e la forma assunta dai legami instaurati. In essa le principali componenti dell'organizzazione possono essere assemblate e riassemblate per far fronte a condizioni competitive complesse e in continuo cambiamento ma soprattutto per saturare la gamma di skill richiesti da una determinata attività.
Il modello basato sulla comunicazione offre numerosi spunti positivi: flessibilità, efficienza, ripartizione del rischio tra le imprese della filiera, possibilità per il top management di concentrarsi su questioni strategiche, controllo del prodotto e dei processi, limitazione degli sprechi e razionalizzazione dell'uso delle risorse, contenimento dello sviluppo verticale della struttura organizzativa. Inoltre tra i possibili effetti dell'implementazione di un modello a rete vi è anche il fatto che l'esistenza di un profondo legame di fiducia all'interno della rete implica la capacità effettiva del collaboratore di dare quanto gli viene richiesto e, dall'altro, la capacità di chi affida le attività, di controllare e coordinare le varie esecuzioni. A quest'ultima questione abbiamo dedicato, in seguito, un'ampia parte del nostro studio.
L'anello debole legato all'utilizzo di tale modello è la metodologia di assegnazione delle risorse di rete ai vari task. Altri problemi riguardano la proprietà intellettuale delle conoscenze acquisite nella rete, le questioni inerenti alla sicurezza che possono spingere l'impresa a non sfruttare adeguatamente le capacità dei partner, il rischio che l'affidamento all'esterno dello sviluppo di certe idee e capacità possa minare il potenziale di sviluppo di lungo termine. Infine, occorre tener conto dei problemi derivanti dal gestire un sistema di relazioni tra imprese indipendenti con propri obiettivi, proprie risorse e forme di organizzazione: a parte i costi del coordinamento, che possono risultare elevati soprattutto se la rete ha molti nodi, la gestione può essere difficile.
E' riconosciuto quindi che la performance di un individuo in un gruppo dipende non solo dalle sue competenze ma anche dalle relazioni che egli ha con gli altri membri del gruppo. Può essere possibile utilizzare alcune sinergie tenendo presente esplicitamente le notazioni topologie dei social network.
Assodato che la struttura di rete di una organizzazione impatta su vari comportamenti collettivi, nel corso del nostro lavoro, si è scoperto anche che tale struttura ha una grossa incidenza sul costo dell'operazione di team formation. Una caratteristica chiave del team formation è rappresentata proprio dalle topologie dei social network; infatti esse descrivono l'interazione diretta tra gli individui di un'organizzazione.
I modelli tradizionali dei fenomeni sociali sono stati riassunti in topologie di rete semplici, o totalmente connesse, ma recenti risultati hanno mostrato che le reti del mondo reale hanno una struttura molto differente e molto più articolata.
Nel nostro lavoro esamineremo infatti anche gli effetti di varie strutture di network sulle dinamiche del team formation in un'organizzazione.

2.4.1 Il social networking

Negli anni le organizzazioni aziendali sono sempre più pensate come un sistema aperto, in cui vengono fortemente considerate le relazioni esistenti tra gli elementi dell'organizzazione e tra questa e l'ambiente esterno; tali relazioni poi vengono caratterizzate e finalizzate allo scopo comune. Quindi le risorse umane non verranno più valutate come parte di un mondo chiuso (l'azienda di appartenenza), ma come una pedina di uno scacchiere, in continua formazione e mutamento. L'elemento attorno al quale ruotano le risorse non è più l'azienda, ma l'obiettivo, il progetto finale!
L'HRM non sarà più un insieme di attività semplicemente a supporto delle direzioni aziendali. I processi di valutazione della performance, di gestione dei talenti, gestione dell'apprendimento sono

tutti elementi responsabili dell'implementazione dei cambiamenti, perciò devono integrarsi in un'unica strategia, alimentandosi tra loro e facendo capo ad una sola base dati, meglio se *sociale*. Gli argomenti della collaborazione sociale e del team formation si sono guadagnati grosso rilievo sul web. C'è un interesse crescente nella comunità per comprendere il processo della collaborazione sociale, ossia, come le persone possano usare le connessioni per formare i team in ambiti lavorativi.
Solo per fare un esempio, le tecnologie web, social e mobile abilitano un processo di selezione continuo e in rapida evoluzione. Oggi chi seleziona può avere un'idea più precisa degli skill e delle figure necessarie attraverso un veloce scambio di informazioni con le varie linee di business; egli può assumere sul web molte più informazioni sul candidato e metterle in relazione tra loro facendo ricerche più mirate su social media orientati al job come *Google+*, *LinkedIn*, *Italian developers*, *Inarcommunity*, *Facebook*, *Pinterest*, portali dedicati o forum professionali, scovando anche candidati interessanti ma poco portati a proporsi attivamente.
E' sempre più evidente che, in una organizzazione, un progetto di successo fa affidamento non solo sulla partecipazione di membri esperti ma anche sulla comunicazione e sulla collaborazione tra essi.
Se vogliamo formare, all'interno di una rete, un team di esperti per un dato task, che consiste in una serie di abilità richieste, non è banale trovare un insieme di persone che soddisfino skill professionali e che siano abili a comunicare efficacemente con ogni altro professionista.
Il precursore di molte teorie sul tema del Team Formation e della formulazione di alcuni modelli è senza dubbio Theodoros Lappas, il quale è stato il primo a considerare, nei suoi studi la struttura sociale della rete che interconnette gli utenti (Cap. 3.4). Egli osservò una forte correlazione positiva tra le attività di un progetto e la natura sociale del team corrispondente. Inoltre, grazie al proliferare del fenomeno del Web 2.0, un numero crescente di siti Open Source Software sono diventati 'sociali', assorbendo, per i loro usi, le caratteristiche del social-networking, quali messaggistica, formazione di community, collegamenti diretti con altre risorse. Ci resta da studiare come si può facilitare il processo di team formation in questa nuova generazione di reti, progettando specifici modelli in grado di considerare le connessioni sociali dei membri nel team. Specificatamente, in questo lavoro, poniamo la seguente questione: *Come costruire un team di utenti efficiente che soddisfi i requisiti di un progetto e allo stesso tempo i membri del team selezionato siano socialmente collegati?*
Per centrare tale obiettivo utilizziamo il concetto di "rete sociale".
Data una rete sociale di esperienze, la disciplina del Team Formation si prefigge il compito di cercare una squadra di esperti per un dato task che a sua volta è costituita di un insieme di abilità richieste. Perché utilizzare proprio la struttura di un Social Network?
Una **rete sociale**, per definizione, consiste di un qualsivoglia gruppo di individui connessi tra loro da diversi legami sociali. Per gli esseri umani i legami vanno dalla conoscenza casuale, ai rapporti di lavoro, ai vincoli familiari. Se il capitale umano è costituito da capacità, conoscenze e competenze, il social capital è funzione delle relazioni che le persone attivano, di cui sono parte; non trascuriamo, inoltre, il fatto che il tipo di relazione che l'azienda supporta inciderà sul rispetto per la persona e il suo coinvolgimento emotivo (Lengnick-Hall, 2009).
Anche se soprattutto nei progetti software, è richiesta assoluta collaborazione tra sviluppatori su diversi livelli di produttività, il team formation non è comunque limitato solamente al mondo dello sviluppo software o delle tecnologie in generale. I team sono fondamentali dovunque ci sia bisogno di una percezione partecipatoria, ossia dove una comunità di utenti contribuisce ad informazioni derivanti da dispositivi sensoriali (telefoni cellulari o dispositivi portatili in genere). Ad esempio tale nozione è stata introdotta nella gestione del traffico cittadino, per la sicurezza sulle piste ciclabili, per la creazione di mappe geograficamente localizzate, nel settore dell' insegnamento a distanza etc... I partecipanti ad un sensing task hanno bisogno di collaborare per la suddivisione del lavoro. In questo contesto, il social network rappresenta il loro grado di cooperazione e perciò, identificando la comunità maggiormente efficiente di utenti per un sensing task si arriva a trovare il miglior team nella loro rete sociale. Crediamo pertanto che il problema di trovare team di utenti su un social

network (per un task collaborativo), ha applicazioni di vasta portata al di là di quelle qui menzionate e sia perciò un importante problema da studiare.
Nel nostro lavoro consideriamo il problema del social team formation in cui l'obiettivo è quello di selezionare un team di utenti da un social network ed effettuare una corretta suddivisione dei task tale che i membri del team siano "socialmente collegati" e nessun utente sia sovraccaricato dalle assegnazioni.
La ricerca condotta nell'ambito di diversi approcci disciplinari ha evidenziato come le reti sociali operino a più livelli e svolgano un ruolo cruciale nel determinare le modalità di risoluzione di problemi. Inoltre esse operano efficacemente anche sui sistemi di gestione del personale, offrendo la possibilità ai singoli individui di raggiungere i propri obiettivi.
Il successo di un progetto dipende, non solo dalla bravura e dall'esperienza delle persone coinvolte, ma anche da come esse realmente sono capaci di partecipare, di comunicare e quindi di lavorare insieme. Solo così si può pensare di costituire poi un team!
Non per ultimo abbiamo scelto di rappresentare il nostro problema attraverso lo strumento del social network anche perché l'analisi delle reti sociali, ovvero la mappatura e la misurazione delle reti sociali, può essere condotta con un formalismo matematico usando la teoria dei grafi.

2.5 Nozioni, vincoli, performance e costi

Nel problema del team formation, è richiesto di trovare un gruppo di utenti che possano soddisfare i requisiti di un task collaborativo, ossia attività che necessitano di più figure professionali possibilmente eterogenee. Esempi di task collaborativi possono essere lo sviluppo di prodotti software oppure i diversi aspetti partecipativi della creazione di conoscenza. Ai membri del team, spesso, è richiesto di lavorare su una base cooperativa indirizzata allo specifico task. I precedenti studi hanno indicato che la cooperazione diviene effettiva in presenza di connessioni sociali. Perciò, la selezione effettiva del team richiede che i suoi membri debbano essere socialmente collegati così come ci debba essere una equa suddivisione dei task tra i membri del team tale che nessun utente sia sovraccaricato dall'assegnamento. Nel nostro lavoro infatti, come già ampiamente sciorinato, indagheremo soprattutto su come questi team possano essere formati al di sopra di un social network.
Dato un insieme di skill richiesti, un insieme di esperti che hanno esperienze in uno o più skill, e un social network (fatto da professionisti), il team formation problem consta quindi nell'identificare un team competente ed altamente collaborativo.
Analizziamo il team-formation nel contesto di un'azienda (all'interno di una rete di aziende) che vuole formare team multipli composti dai suoi membri. L'organizzazione deve suddividere i suoi membri in team, ciascuno sarà costituito da specifici profili, ognuno dei quali è associato ad una differente retribuzione. Connettere un membro ad ogni altro in un social network da garanzie sul fatto che essi possano aiutarsi vicendevolmente al di fuori del task d'interesse, innalzando le probabilità di successo del progetto.
Questo problema, introdotto da Lappas (2009) nel contesto del social network, ha attratto recentemente interesse nella comunità del data mining. Un suo sotto-problema fin qui studiato con molta attenzione nelle operazioni di ricerca è quello dell'assegnamento: dati un set di agenti e un insieme di task, l'obiettivo è di trovare un assegnamento agente-task che minimizzi il costo dell'assegnamento stesso. Esattamente un agente sia assegnato ad un task ed ogni task sia assegnato a qualche agente.
Esistono già numerosi metodi atti alla risoluzione della questione che considerano però assunzioni troppo restrittive al punto che, spesso, non possono essere implementate in scenari realistici.
Una iniziale modellazione indirizza il team formation verso un problema di matching con peso massimo in un grafo bipartito pesato (Figura 1). In esso i nodi del grafo saranno suddivisi tra nodo-user e nodo-task, ottenendo una bi-partizione del grafo originario. Un nodo-user, che conterrà uno o

più skill, può essere connesso a 0 o più nodi-task e viceversa. Le connessioni rappresenteranno il grado di partecipazione al task da parte dell'utente.

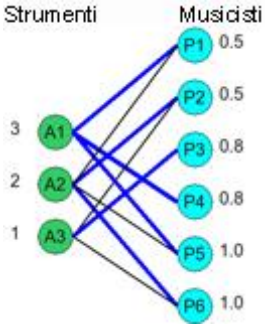

Figura 1: Classico problema di matching all'interno di un'orchestra musicale.

Come accennato, in contrasto con il problema di assegnazione, il problema del team formation considera anche il social network su cui poggia. Questo ci consente, per esempio, di tener conto delle precedenti collaborazioni tra gli esperti. Uno dei vantaggi dell'uso di un social network è che, circa i team che hanno lavorato insieme precedentemente, ci si aspetta di avere costi di comunicazione più bassi, e quindi un risultato migliore. Discutendo poi anche delle topologie di rete, sarà possibile ottimizzare quest'aspetto.

I criteri che misurano l'efficienza dei team, che si trovano in letteratura oggi, sono basati sui cammini minimi, sul costo del minimum spanning tree del sottografo indotto dal team, sulla taglia complessiva del team, sul diametro della rete, sul concetto di densità del grafo, etc…

Ognuno di questi parametri potrà essere modellato come un costrutto matematico, divenendo così un opportuno vincolo nella costruzione del modello per la risoluzione del problema del team formation. La selezione dei vincoli è fondamentale per evitare che la formulazione si distacchi troppo dal mondo reale e diventi applicabile solo nella teoria.

L'obiettivo del nostro lavoro è quello di considerare il problema del team formation in uno scenario più realistico possibile e presentare una formulazione innovativa basata sulla generalizzazione del problema tenendo conto dei vincoli reali e quindi anche di costi effettivi, comprendendo per esempio la possibilità di aggiungere o eliminare nodi dalla rete, includere o no i decision maker/team leader, considerare il concetto di località o prossimità sociale degli user…

Lappas inizialmente considerò il problema del team formation su di un singolo task con l'obiettivo di minimizzare il costo di coordinazione, ma ignorando lo scopo del bilanciamento dei carichi di lavoro. Anagnostopoulos (2010) considerò invece la formazione di team con l'obiettivo di bilanciare il carico di lavoro tra gli utenti, ma ignorò i costi di comunicazione…

Gli algoritmi maggiormente efficienti saranno pertanto quelli che formano team che soddisfino sempre gli skill richiesti, forniscano garanzie di approssimazione rispetto ai vari costi, e siano competitivi rispetto al bilanciamento dei carichi, grazie alle informazioni ricavate dalla comunicazione tra i nodi della rete.

Considerando quanto già detto sul social networking, sul concetto generale di team formation, nonché sui vincoli, i costi e le performance, ci adoperiamo per introdurre un nuovo modello atto alla risoluzione del team formation che tenga presente tutte le nozioni preesistenti, da noi ritenute opportune, con l'introduzione di altre notazioni derivanti ad esempio dal concetto di fiducia tra chi assegna le risorse e gli utenti stessi (*trust*).

3. Formazione di gruppi: Strumenti e Modelli

Per costruire un buon strumento di ausilio al Team Formation occorre strutturare un modello fondato su esperienze reali, analitiche o su teorie prodotte, inoltre occorre utilizzare con sapienza gli strumenti messi a disposizione dalle discipline scientifico-tecnologiche.
In questa sezione, dopo una brevissima introduzione sulle notazioni fondamentali, sulle strutture dati (diagrammi, matrici e grafi) e circa i rudimenti di un generico modello matematico risolutivo, vedremo alcuni modelli, reperibili nel panorama dell'attuale stato dell'arte, che offrono interessanti spunti di riflessione.
L'approccio analitico riportato nel paragrafo 3.2 traduce in modello il concetto ingegneristico di team formation. La squadra, secondo gli ideatori Zakarian e Kusiak del Department of Industrial Engineering, University of Iowa (1999), verrà costituita in funzione del tipo di prodotto o servizio che deve essere sviluppato, in base alle richieste del cliente, all'ingegnerizzazione, alle specifiche caratteristiche e da altro ancora. La metodologia sviluppata è strutturata in modo tale da scindere gli aspetti pratici da quelli non tangibili circa la formazione di team multi-funzionali. E' basata sul Processo di Gerarchia Analitica, ossia un approccio multicriterio di decision-making che fornisce una piattaforma di problem-solving, e una procedura sistematica per rappresentare gli elementi di ogni problema. Le caratteristiche più importanti dell'Analytical Hierarchy Process sono la sua semplicità, la robustezza e la sua abilità di includere alcuni particolari processi nella fase di decision-making. Discuteremo l'applicazione della QFD (funzione di qualità) al fine di costruire un framework concettuale che avrà lo scopo di dare un valore ai membri di un team, basandosi sui requisiti e sulle caratteristiche del prodotto.
Nel paragrafo 3.3, usando un semplice modello computazionle agent-based, sono stati condotti vari esperimenti virtuali per esaminare l'impatto delle strutture delle reti complesse sulle dinamiche del team formation. Matthew Gaston e Marie desJardins dell'University of Maryland (2005) hanno analizzato quattro differenti topologie di rete con le relative differenti strutture organizzative che esse richiedono. Attraverso tecniche di simulazione, essi forniranno i risultati sotto forma di performance relativa al team formation.
Il modello introdotto nel paragrafo 3.4 è il primo in cui si fa riferimento ai social network come infrastruttura per il Team Formation. Theodoros Lappas (2009) richiede che i membri del team non solo conoscano i requisiti del task, ma possano anche lavorare efficientemente insieme, sfruttando le caratteristiche del social. L'efficienza del metodo è misurata usando due differenti funzioni di comunication-cost, giungendo alla conclusione che entrambi sono NP-hard. Analizzando i legami di questo metodo con i problemi combinatori esistenti, sono introdotti poi nuovi algoritmi per la loro risoluzione.
Il modello presentato nel paragrafo 3.5 propone un'approssimazione greedy per il problema dell'allocazione di membri a squadre, tenendo conto della struttura del social network. Das e Szymanski del Rensselaer Polytechnic Institute of Troy di New York (2012) dimostrano che l'approssimazione è vicina all'ottimo nei problemi dove l'allocazione ottima non può essere esplicitamente calcolata. L'idea alla base di tale strategia è quella di caratterizzare il livello di abilità di ogni membro con la probabilità di completare con successo un particolare tipo di task. Tale approssimazione fornisce benefici significativi sull'allocazione ottima se non viene presa in considerazione la struttura della rete, specialmente su reti molto grandi. E' trattato anche il concetto di topologia della rete, focalizzandosi su quelle che massimizzano la performance del team. Tali analisi sono state caratterizzate anche dalla struttura del grafo, dalla connettività, e dalla taglia del team.
Nel paragrafo 3.6 presentiamo uno dei più completi lavori sul Team Formation. Anagnostopoulos et all. (2002) hanno studiato il problema del Team Formation Online, considerando uno scenario in cui

le persone posseggono differenti skill e la compatibilità tra i potenziali membri del team è modellata da una social network. La sequenza di task giunge via rete e ognuno di essi necessita di uno specifico set di skill. L'obiettivo è quello di formare un nuovo team finalizzato all'esecuzione di ogni task, tale che ogni team possegga tutti gli skill richiesti dal task. E' proposto poi il primo algoritmo online che assembla i team in relazione ai task, tenendo conto dei costi di coordinamento e di bilanciamento del carico di lavoro.
L'obiettivo del paragrafo 3.7 è quello di considerare il problema del team formation in un setting realistico e presentare una nuova idea basata su sottografi densamente popolati. La formulazione ideata da Rangapuram, Bühler ed Hein della Saarland University di Saarbrücken (2013), consente la modellazione di molti aspetti naturali quali: l'inclusione di team leader designati e/o di gruppi di esperti, la diminuzione della taglia o più generalmente del costo del team e l'introduzione di vincoli sulla *località* del team (in senso geografico o sociale). Il nuovo metodo (FORTE) nasce dallo studio del problema classico del sottografo denso con cardinalità vincolata (DSP). Esso si basa sulla soluzione di una relazione equivalente continua del DSP generalizzato.
Il paragrafo 3.8 prende spunto dai lavori di Datta e Majumder della Alcatel-Lucent (2012) che per la prima volta, nel contesto del team formation, tengono conto dei vincoli di capacità. Ossia, per un dato task, l'obiettivo diventa quello di trovare un team di utenti che, oltre ad essere 'socialmente efficienti', siano oggetto di una suddivisione equa dei compiti tale che nessun utente venga sovraccaricato dall'assegnamento e soprattutto a nessun utente siano assegnati task al di là delle proprie capacità. Vengono effettuate delle ricerche su particolari siti Open Source (per la prima volta su GitHub) che, grazie al proliferare del fenomeno del Web 2.0, sono diventati 'sociali', assorbendo, per i loro usi, le caratteristiche del social-networking. Ci resta da studiare come si può facilitare il processo di team formation, tenendo conto dei vincoli di cui sopra, in questa nuova generazione di reti, progettando algoritmi specializzati per il team selection in grado di considerare le connessioni sociali dei membri nel team.
Per ognuno di questi modelli sarà brevemente illustrata una parte storico-motivazionale, seguita dall'elencazione delle notazioni matematiche, nonché dalla spiegazione del modello vero e proprio. In genere da ognuno di questi modelli teorici nascono uno o più algoritmi che verranno analizzati e ne saranno calcolate le relative complessità temporali. All'occorrenza saranno introdotti i vari ipotetici utilizzi applicativi nonché i risultati sperimentali con le relative performance sotto forma di grafici, ove è possibile.

3.1 Notazioni e Strutture dati

3.1.1Notazioni

Formalizziamo il problema esposto nella sezione precedente come segue.
Dato un task T, un insieme di esperti V con skill multipli e un grafo G(V,W) che descrive le "connessioni" tra gli esperti, il team formation può essere quindi definito come il problema dell'identificazione del team $C \subseteq V$ che sia capace di eseguire il task T e contemporaneamente i suoi membri siano compatibili alla collaborazione.

3.1.2 Diagrammi e Matrici

Per ottenere risultati certi è bene fare uso di tool basati su strumenti utili all'assegnazione delle responsabilità come i seguenti.

- Diagrammi gerarchici:

- **WBS** *(Work Breakdown Structure)*: restituiscono la struttura analitica del progetto con l'elenco di tutte le attività ed identificano il responsabile di ognuna.
- **OBS** (*Organization Breakdown Structure*): struttura gerarchica organizzata per funzioni, dipartimenti e persone che saranno coinvolte nel progetto.
- **RBS** *(Resource Breakdown Structure)*: struttura gerarchica organizzata per tipologia di risorsa, utile anche per tracciare i costi.

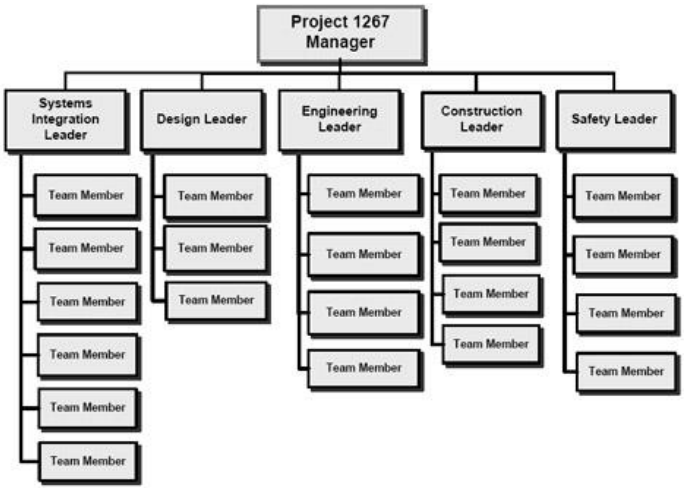

Figura 2: Esempio di OBS - Organization Breakdown Structure

- Diagrammi a matrice:

- **RAM** *(Responsibility Assignment Matrix)**:*** usata per illustrare le connessioni tra il lavoro necessario e i membri del Team. Può essere vista come incrocio della WBS e della OBS.
- **RACI** (Responsible, Accountable, Consult and Inform): detta anche *Matrice della Responsabilità*, è un particolare formato di RAM che mostra anche il grado di coinvolgimento ai vari livelli.

RACI Chart	Person				
Activity	Ann	Ben	Carlos	Dina	Ed
Define	A	R	I	I	I
Design	I	A	R	C	C
Develop	I	A	R	C	C
Test	A	I	I	R	I

R = Responsible A = Accountable C = Consult I = Inform

Figura 3: Esempio di RACI - Matrice della Responsabilità

Utilizzando tali strutture, si otterranno principalmente tre grossi vantaggi: miglioramento delle competenze dei singoli, aumento dell'efficienza complessiva del gruppo e sensibile diminuzione del turn-around tra i membri.
Nel paragrafo 3.2 vedremo come l'utilizzo di una particolare matrice di qualità possa fornire elementi fondamentali per l'analisi e la formazione del team.

3.1.3 Teoria dei grafi e matrice d'efficienza

Per costruire un modello risolutivo al nostro problema sono possibili vari approcci. Tutti o quasi fanno riferimento ad una rete di utenti o di conoscenze spesso traducibile nella struttura di un grafo.

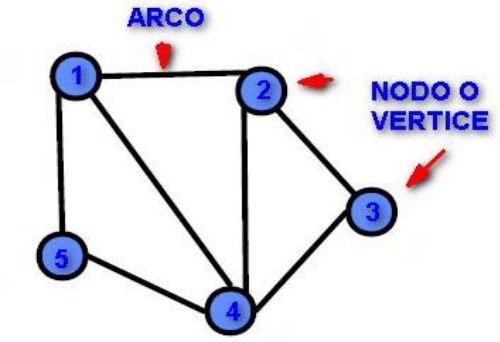

Figura 4: Grafo

La teoria dei grafi è un comodo strumento per la definizione e la formalizzazione di numerosi problemi di ottimizzazione. Spesso infatti è utile, non tanto cercare la soluzione di un dato problema, bensì ricondurlo ad una classe di problemi noti dei quali si conosce, o almeno si ha idea, dei possibili approcci per la soluzione.
Inoltre, il linguaggio dei grafi consente di rappresentare in modo semplice la struttura di molti problemi applicativi, quindi la soluzione trovata potrà essere con facilità estesa ad altri eventuali progetti. Definiamo ora un grafo.
Un *Grafo* G(V, E) è una coppia di insiemi V e E. Gli elementi dell'insieme V sono detti vertici, e gli elementi dell'insieme E sono detti archi o connessioni e rappresentano le relazioni che esistono tra i vertici. In un grafo diretto o digrafo le connessioni sono descritte da coppie ordinate di vertici e sono rappresentate con $E_k = (v_i, v_j)$, mentre in un grafo indiretto sono descritte da coppie non ordinate di vertici e sono rappresentate con $E_k = \{v_i, v_j\}$.

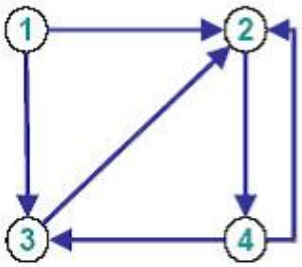

Figura 5: Grafo diretto

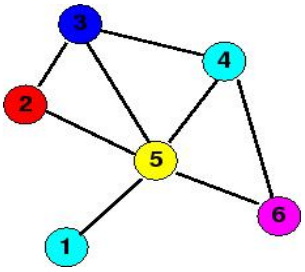

Figura 6: Grafo indiretto

Una connessione si dice incidente su un vertice, quando su tale vertice ha uno dei due punti di collegamento. Il grado di un vertice è il numero di connessioni su di esso incidenti. Un ipergrafo è una estensione di un grafo nel quale le connessioni possono essere incidenti su un numero qualsiasi di vertici. Una *cricca* (o clique) è un insieme V di vertici in un grafo non orientato G, tale che, per ogni coppia di vertici in V, esiste un arco che li collega.
Visto che ne faremo largo uso, sembra giusto introdurre il concetto di *densità* che letteralmente esprime il livello generale dei legami tra i nodi in un determinato grafo. Vi sarà maggiore densità quanto più numerose sono le linee direttamente collegate. La densità dipende da:
- inclusività: numero totale dei punti collegati meno il numero dei punti isolati;
- grado di connessione: alcuni nodi avranno collegamenti con molti altri, mentre altri punti avranno connessioni meno numerose. Quanto più i nodi (n) hanno elevate connessioni, tanto maggiore sarà la densità del grafo.

$$densità = \frac{l}{n(n-1)/2}$$

Introducendo il concetto di "archi pesati" in un grafo, generici problemi possono essere ricondotti alla categoria dei "Problemi di cammino minimo e massimo". Si definisce grafo pesato un grafo ai cui nodi e/o archi sono associati valori numerici detti pesi. Nel nostro caso i pesi sugli archi possono rappresentare parametri quali i costi.

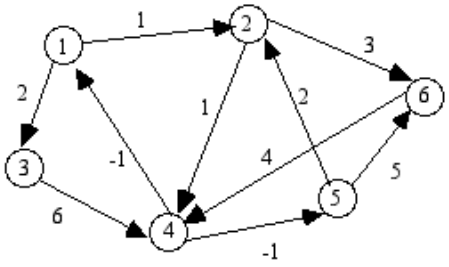

Figura 7: Grafo pesato

Gran parte dei problemi rappresentabili con un grafo si riducono alla risoluzione del problema del cammino a costo minimo e a costo massimo per i quali in letteratura sono stati proposti molti algoritmi che ricercano, con diversi approcci, la soluzione migliore.
Il problema del cammino minimo (quello a cammino massimo può essere considerato una sua estensione) può essere formalizzato da un grafo diretto connesso con archi pesati ed un vertice, chiamato sorgente, su cui non insistono archi in ingresso.
Si deve ricercare nel grafo il percorso dal vertice sorgente a un altro qualsiasi vertice che abbia la somma minima dei pesi delle connessioni. Di seguito indicheremo con W= $\{w_1, w_2, \dots, w_n\}$ l'insieme dei numeri reali che rappresentano i pesi delle connessioni.

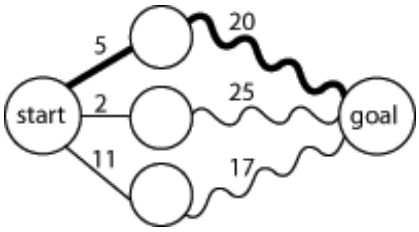

Figura 8: Modello del cammino a costo minimo

In figura 8 è illustrato un particolare grafo G(V, E, W) che modellizza un problema a cammino minimo con un vertice definito "start" che denota la sorgente e un vertice etichettato "goal" che rappresenta la destinazione (o pozzo). Sapendo che i grafi con cicli negativi sono quelli in cui se due vertici appartengono a un ciclo di costo negativo, non esiste nessun cammino minimo finito tra di essi, il nostro grafo deve avere la caratteristica di esser privo di cicli negativi, altrimenti il problema è inconsistente. Normalmente in un grafo che rappresenta una rete di flusso ogni vertice giace su di un qualche cammino dalla sorgente al pozzo, cioè esiste sempre un cammino che collega due vertici qualsiasi. Il grafo è quindi connesso e si ha $|E| \geq |V| - 1$.
Come spiegato ampiamente in questo lavoro, i problemi di assegnazione fanno parte dei problemi lineari, essi sono quei problemi di ricerca operativa in cui bisogna assegnare diverse attività in maniera ottimale. Il problema di assegnamento è considerato un problema combinatorio e generalmente questi problemi vengono risolti grazie all'attribuzione di *n* risorse ad *n* destinazioni di modo che ci sia una corrispondenza biunivoca tra la *n*-esima risorsa e l'*n*-esima destinazione. Per tale motivo è definita una particolare struttura dati che raccoglie i dati relativi a risorse, task e costi. La tabella dei costi o dei tempi è detta anche matrice dell'efficienza.

	B_1	B_2	…	B_n
A_1	C_{11}	C_{12}	…	C_{1n}
A_2	C_{21}	C_{22}	…	C_{2n}
…	…	…	…	…
A_n	C_{n1}	C_{n2}	…	C_{nn}

Tabella 2 - Matrice dell'efficienza

dove:
A_i indicano le risorse
B_k indicano i task
C_{ik} indicano i costi (o i tempi)

3.1.4 Il Modello generico

Per l'impostazione del modello matematico si fa riferimento ad una variabile booleana X, la quale assume valore 0 in caso di non assegnazione e 1 in caso di assegnazione. Il generico modello matematico di rappresentazione può essere così descritto dalla seguente formula.

$$Z = \sum_{i=1}^{n} \sum_{k=1}^{n} c_{ik}\, x_{ik} \qquad \textit{da minimizzare}$$

con i seguenti vincoli

$$\begin{cases} \sum_{i=1}^{n} x_{ik} = 1 \\ \sum_{k=1}^{n} x_{ik} = 1 \\ x_{ik} \geq 0 \end{cases} \qquad \textit{con i e k = 1,2,3,4, ...n}$$

Si ha quindi la seguente tabella:

	B_1	B_2	…	B_n
A_1	X_{11}	X_{12}	…	X_{1n}
A_2	X_{21}	X_{22}	…	X_{2n}
…	…	…	…	…
A_n	X_{n1}	X_{n2}	…	X_{nn}

Generica formulazione per il problema dell'assegnazione.

Visto che la nostra specifica problematica rientra nella classe dei problemi a cammino minimo e massimo, è possibile trovare il percorso desiderato di peso minimo tra sorgente e destinazione nel nostro grafo bipartito, applicando n volte algoritmi noti quali l'algoritmo *Dijkstra* oppure quello di *Ford-Fulkerson* ma con risultati non ottimali.
Per tale motivo analizziamo, nei seguenti paragrafi, altri modelli presenti nell'attuale stato dell'arte, che considerano strategie a volte adiacenti a quelle note altre volte molto distanti per valutare quanto i relativi risultati differiscano dall'ottimo.

3.2 Approccio analitico

La creazione di team multi-funzionali può essere considerata come la chiave di risoluzione per il problem solving. Il concetto della formazione di team polifunzionali, nei recenti anni, ha catturato molta attenzione. Uno dei principali sforzi ingegneristici intorno a questo concetto è stato il *Concurrent Engineering* (CE).
La più grande attività di ricerca, in tale ambito, fu portata avanti da *Zakarian e Kusiak* (1999) che svilupparono una metodologia per la selezione dei team. Tale metodologia è basata sull'approccio analitico (AHP) e su di un metodo qualitativo (QFD).
Essi presentarono un framework concettuale in cui i membri del team venivano scelti in base ai requisiti e alle caratteristiche di ciò che si voleva produrre, nonché alla possibile ingegnerizzazione dello stesso prodotto.
Inoltre, per determinare la composizione ottima di un team, il modello di programmazione matematica utilizzato per giungere a tale risultato faceva riferimento all'applicazione di una distribuzione della funzione di qualità (QFD).

3.2.1 La funzione di qualità

QFD (Quality Function Deployment), già nota da alcuni decenni (Saaty, 1992), in generale fornisce un approccio sistematico allo sviluppo di un qualsiasi prodotto. In particolare, QFD identifica i

requisiti generali che un nuovo prodotto o servizio deve soddisfare, in base alle preferenze degli utenti. L'essenza di QFD è quella di traslare le richieste dei consumatori nella progettazione del servizio tale da ottenere una produzione di alta qualità. I requisiti richiesti dagli utenti sono il fondamento del processo e vengono riportati nella matrice QFD che identifica le relazioni tra tali requisiti richiesti e ciò che deve essere sviluppato.
Vediamo come viene utilizzato il concetto di QFD al fine di rappresentare il modello di selezione per un team multifunzionale.
L'applicazione del concetto di qualità QFD per la formazione del team muti-funzionale inizia considerando come attori principali i project managers e gli utenti.
Essi sviluppano una matrice QFD, nella quale sono relazionati i requisiti richiesti con le caratteristiche d'ingegnerizzazione (*requirements*) del prodotto (Figura 9(a)).
I requirements sono rappresentati anche nella seconda matrice in cui le righe sono le *Engineering characteristics* mentre sulle colonne avremo i membri del team di sviluppo (Figura 9(b)). In altre parole, la planning matrix succitata mette in relazione le caratteristiche ingegneristiche del prodotto con i potenziali membri del team che ipoteticamente potrebbe garantire tali caratteristiche.

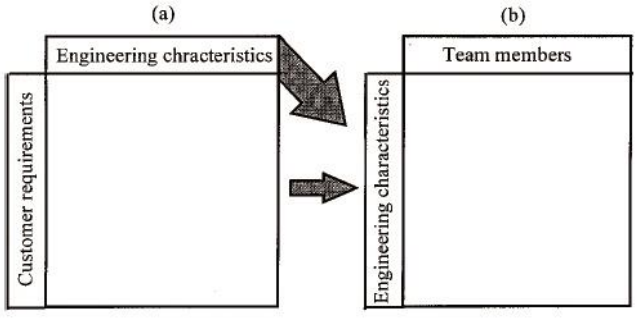

Figura 9: (a) Matrice QFD Customer requirements/ Engineering characteristics;
(b) Engineering characteristics/Team members.

Le matrici sopra riportate sono usate poi per organizzare i fattori che successivamente verranno considerati nella struttura gerarchica AHP, relativamente alla selezione del team, che vedremo tra poco.

3.2.2 Il Processo Analitico

L'Analytical Hierarchy Process (AHP) è un metodo di decisione multicriterio che usa strutture gerarchiche o reti per rappresentare un problema decisionale. Utilizzando tale struttura e criteri alternativi basati sul concetto di giudizio, AHP sviluppa delle priorità che saranno poi utilizzate dal sistema che si occuperà della fase di decision-making.
Questa metodologia ideata da Thomas Saaty trova numerose applicazioni grazie alla sua semplicità e all'abilità nel risolvere complessi problemi decisionali.
A noi interessa la variante indirizzata al problema del team formation e alla sua ottimizzazione.
AHP è basata sui seguenti tre principi:

1. *Decomposizione del problema* - Il concetto alla base di tale metodologia è decomporre un complesso problema multicriterio in una gerarchia dove ogni livello consiste di pochi elementi gestibili i quali sono poi decomposti in altri insiemi di elementi. L'obiettivo comune verrà di volta in volta posizionato al vertice della gerarchia. Il successivo livello della gerarchia contiene attributi o criteri che contribuiscono alla qualità delle decisioni. Ogni attributo potrebbe essere decomposto in attributi più dettagliati. Il livello più basso della gerarchia contiene decisioni alternative.

2. *Giudizio comparativo* - Dopo aver costruito tale rete gerarchica, si può utilizzare il giudizio in maniera comparativa per determinare le priorità (o misure d'importanza) degli attributi ad ogni livello della gerarchia decisionale.
3. *Sintesi delle priorità* - Per determinare le priorità generali tra le varie alternative possibili si sintetizzano le priorità degli attributi.

3.2.3 L'algoritmo

Il primo step consiste nel costruire una "matrice di paragone" che include gli elementi del Livello 1 (criteri). Poi, comparando a due a due i criteri che rispettano l'obiettivo generale, viene creata una scala di rapporti. L'importanza relativa di ogni criterio è stimata utilizzando il principio dell'"autovettore principale della matrice" di Saaty (1981).
Poi, usando un singolare metodo di paragone a coppie, viene determinata una scala d'importanza relativa delle altre possibili alternative, secondo ogni criterio. La comparazione a coppie tra i criteri (come tra le alternative) nasce dallo sviluppo dei 9 punti utilizzati da Saaty: 1, 3, 5, 7, e 9 per esprimere la scala dei rapporti circa la preferenza tra le alternative; 2, 4, 6 e 8 per le soluzioni di compromesso e per i reciproci (gli inversi).
La tabella 3 riassume l'importanza relativa della scala di Saaty modificata per il problema del team selection.

Scala di valori	Definizioni
1	**Uguale valore** – Due membri contribuiscono nella stessa misura al raggiungimento dell'obiettivo
3	**Moderata importanza di un membro rispetto ad un altro** – L'esperienza e le capacità cognitive favoriscono leggermente un membro del team su di un altro
5	**Importanza essenziale o forte** – Un membro è fortemente preferito ad un altro
7	**Importanza molto forte o dimostrata** – Un membro è decisamente preferito e la sua superiorità è dimostrata nella pratica
9	**Importanza estrema** – L'evidenza secondo cui si preferisce un membro rispetto ad un altro è pari al più alto valore di affermazione
2, 4, 6, 8	**Valori mediani tra due giudizi adiacenti** – Quando c'è bisogno di un compromesso
	Reciproco – Per comparazione inversa

Tabella 3 - La scala di T.L. Saaty applicata al team selection

Tipicamente, il decision-maker effettua la sua valutazione, basandosi sulla triangolazione di una matrice di paragone, in cui, nella parte bassa del triangolo sono posizionati i reciproci. In altre parole se il fattore di scala 3 è assegnato alla entry (*i*,*j*) della matrice, dove *i* e *j* sono rispettivamente la riga e la colonna, allora il valore 1/3 è assegnato alla entry (*j*,*i*). Visto che anche quando compariamo un criterio con se stesso o con l'alternativo assumono la stessa importanza, gli elementi diagonali della matrice sono sempre uguali ad uno.
Il paragone a coppie degli elementi rispetto all'assegnazione diretta dei valori di preferenza è più preciso anche se più complesso.
La metodologia AHP fornisce un indice per misurare ogni inconsistenza di giudizio nelle matrici di paragone sottoforma di gerarchia perfetta. Sapere se un giudizio è consistente o meno è importante per una sua successiva rivalutazione.
La gestione della consistenza nel processo di selezione del team è così definita: se un responsabile preferisce leggermente il membro B sul membro C, e il membro C su D, allora ci si aspetta che egli

preferisca il membro B su D; questo è il più alto ordine possibile di asserzione. L'indice di consistenza (*CI*) e la percentuale di consistenza (*CR*) per una matrice di paragone A sono definiti come segue:

$$CI = (\lambda_{max} - n)/(n - 1), \quad (3.1)$$
$$CR = (CI/ACI)100\%, \quad (3.2)$$

dove λ_{max} è il più grande autovalore della matrice di paragone, *n* è la dimensione della matrice, e *ACI* è l'indice medio dei pesi casualmente generati.
Se il valore calcolato di *CR* per la matrice di paragone è minore del 10%, il giudizio di consistenza di coppia è accettato basandosi sulla regola empirica di Saaty (1986) .
Quando il tasso di consistenza supera il 10%, significa che i giudizi espressi dagli esperti sono considerati inconsistenti, per cui i decision-makers hanno la facoltà di riesaminare tali giudizi.
La preferenza relativa nella gerarchia è ottenuta attraverso il confronto a coppie sull'importanza degli elementi dello stesso livello rispetto al genitore assegnato, su cui un decision-maker o un gruppo di decision-maker esprimono i loro giudizi.
Aczel (1983) dimostrò che il giudizio collettivo deve soddisfare la proprietà reciproca. Questo implica che quando un gruppo usa l'AHP, i singoli giudizi possono essere sintetizzati in un unico giudizio secondo il significato geometrico. Per una discussione dettagliata su come organizzare i gruppi di decisione con AHP, su come dare un grado al gruppo, includendo suggerimenti per l'assemblaggio dei gruppi, le costruzioni delle gerarchie, e come implementare i risultati, è possibile consultare importanti pubblicazioni: Golden (1989), Islei (1991), Saaty (1992), Dyer e Forman (1992).

3.2.4 Il modello

Nel problema di selezione dei team, in genere, AHP viene applicato come di seguito illustrato; in esso i valori che contribuiscono a stabilire i criteri conseguono dal giudizio di vari utenti.
Per selezionare un team, prima bisogna dedurre tutte le possibili caratteristiche o gli attributi desiderati, poi sviluppare un metodo per assegnare la loro importanza secondo l'obiettivo generale. Tutti gli elementi, ossia, i requisiti, le caratteristiche d'ingegnerizzazione e i membri del team ottenuti dalla matrice di planning in Fig 9 (a e b) contribuiscono a formare insieme il modello descritto nella gerarchia in Figura 10.

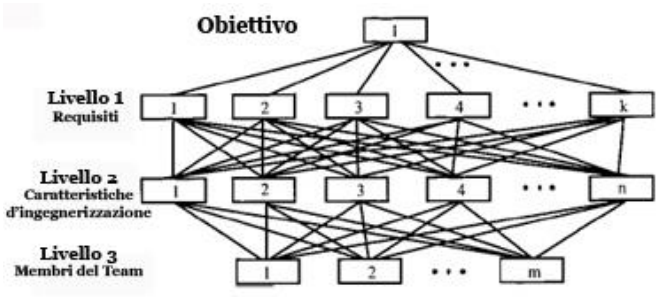

Figura 10: Struttura gerarchia per il problema di selezione del team

Per poter giungere alla designazione dei membri del team bisogna associare delle priorità secondo le caratteristiche d'ingegnerizzazione considerate. In genere una di esse priorizza tutti gli elementi della gerarchia. Successivamente è ottenuta la misura normalizzata della priorità dei membri del team secondo ogni caratteristica del Livello 2.

Ora verrà presentato il modello di programmazione matematica che è usato per determinare la composizione ottimale dei team.
Il problema della formazione di team multi-funzionali è formulato come un modello di programmazione intera. Il modello è basato sulle caratteristiche d'ingegnerizzazione (sul tipo di priorità dei membri sulla matrice d'incidenza). Ogni riga della matrice corrisponde ad una distinta caratteristica d'ingegnerizzazione del prodotto. Ogni colonna denota un tipo di membro del team. Ogni entrata w_{ij} nella matrice d'incidenza indica la priorità (il peso) del membro del team di tipo *j* nel rispetto della caratteristica d'ingegneria *i*. Per formulare il modello sono introdotte le seguenti notazioni:
i = indice per le caratteristiche d'ingegneria;
j = tipo di membro del team;
n = numero delle caratteristiche d'ingegnerizzazione;
m = numero dei tipi di membri;
w_{ij}= peso di priorità del membro del team di tipo *j* rispetto alla caratteristica *i* da ingegnerizzare;
p = numero dei team multifunzionali;
m_j = numero dei progetti che un membro di team tipo *j* può garantire. Il valore di m_j può essere funzione del tempo, della tecnologica, della progettazione etc;
M = numero positivo arbitrariamente grande;

x_{ij} = { 1 se il membro del team di tipo *j* appartiene al team che fornisce la caratteristica *i* d'ingegneria; 0 altrimenti.

y_i = { 1 se il team per la caratteristica d'ingegneria *i* è formato; 0 altrimenti.

La funzione obiettivo da massimizzare sarà:

$$max \quad \sum_{i=1}^{n} \sum_{j=1}^{m} w_{ij} x_{ij} \ , \qquad \text{(v1)}$$

vincoli:

$$\sum_{i=1}^{n} x_{ij} \leq m_j \qquad j = 1, \dots, m \ , \qquad \text{(v2)}$$

$$\sum_{i=1}^{n} y_i \leq p \ , \qquad \text{(v3)}$$

$$\sum_{j=1}^{m} x_{ij} \leq M y_i \qquad i = 1, \dots, n \ , \qquad \text{(v4)}$$

$$x_{ij} = 0,1 \qquad i = 1,\dots, n \ \text{ e } \ j = 1,\dots, m \ , \qquad \text{(v5)}$$

$$y_i = 0,1 \qquad i = 1, \dots, n \ . \qquad \text{(v6)}$$

Il vincolo (v2) impone il limite superiore sul numero dei progetti che un membro del team di tipo *j* è capace di intraprendere.
Il vincolo (v3) specifica il numero delle squadre di lavoro richieste.
(v4) garantisce che un membro del team di tipo *j* appartiene al team *i* solo quando il team *i* è già stato formato.
(v5) e (v6) garantiscono l'interezza .
Nel modello ((v1)-(v6)) possono essere facilmente incorporati dei limiti aggiuntivi, come ad esempio informazioni sul budget disponibile, sulla taglia dei requisiti dei team, etc..

3.3 Il modello Agent-based

Alcuni modelli sono utili per meglio comprendere come e quanto la struttura dati sia fondante nel processo di ottimizzazione dell'assegnazione dei task alle risorse.
In questo paragrafo, per esplorare gli effetti delle strutture di rete sul team formation, è proposto un semplice *modello organizzazionale agent-based*.

3.3.1 Il modello

Il concetto base prevede che i task vengano generati e lanciati sulla rete in modo tale che tutti gli agenti (le risorse dell'organizzazione) ne recepiscano l'avviso di pubblicizzazione e si adoperino, secondo il modello che vedremo, per costituire i team che completino i succitati task.
Alla costituzione di questi team possono partecipare tutte le strutture di rete che contengono gli agenti in grado di recepire il messaggio di avviso. I task sono generici poiché essi richiedono solo che venga costituito un team di agenti con gli skill necessari alla realizzazione di uno specifico task, ma non individuano alcuna risorsa. Una volta che i requisiti di uno skill per un dato task sono tutti soddisfatti, gli agenti implicati rendono il team attivo.
L'organizzazione è rappresentata da un grafo $G=(V, E)$ contenente N nodi che rappresentano gli N agenti. L'insieme degli agenti è $A = \{a_1, a_2, \ldots, a_N\}$, dove ogni agente a_i è situato nel corrispondente nodo v_i nel grafo G, e N=|V|.
Ad ogni agente è assegnato uno dei tre valori, $s_i \in \{0,1,2\}$, ai quali corrisponde rispettivamente uno stato: *non impegnato, impegnato* e *attivo*. Un agente non impegnato è disponibile e non assegnato ad alcun task. Un agente impegnato ha scelto un task, ma il team per lo specifico task non è ancora completamente formato. In fine, un agente attivo è un membro di un team che ha assegnato tutti gli skill richiesti per un task e lavora attivamente su quel task. L'agente attivo diventa non impegnato, e quindi disponibile, quando il task è completato. Un task è pubblicizzato mediante un avviso (o annuncio). Ogni annuncio di pubblicizzazione ha un *time to live*, raggiunto il quale, se non sono stati ancora assegnati gli skill richiesti per un dato task, gli agenti transitano dallo stato impegnato a non impegnato.
Gli agenti possono anche essere caratterizzati da un singolo skill, $\sigma \in [1, \sum]$, dove $\sum$ è il numero dei differenti tipi di skill che sono presenti nell'organizzazione. L'insieme degli archi E contiene le interazioni tra gli agenti. I task invece vengono introdotti ad un rapporto fissato μ. Ogni task K ha una taglia di requisiti associata |K|, e un vettore |K|-dimensionale di skill richiesti R_k. Gli skill richiesti per un dato task K sono scelti in modo omogeneo da $[1, \sum]$. Se tutte le richieste del task non possono essere soddisfatte, per garantire che le risorse assegnate al task siano liberate, è stabilito che ogni task sia annunciato per un numero finito di volte proporzionalmente alla sua taglia. Tale taglia è chiamata $\delta|K|$, dove δ è un parametro di modello.
Similmente, i team che devono eseguire le richieste del task dato sono attivi solo per un numero finito di volte (chiamato $\alpha|K|$, dove α è un parametro di modello).

Figura 11 : Network agent-based.

Gli agenti vengono assegnati ai task secondo un preciso set di regole. Questa è una fase molto delicata ed incisiva nell'intero processo di assegnazione. E' banale infatti dimostrare che le interessanti dinamiche di questo modello di team formation dipendono, in gran parte, dalle strategie che gli agenti impiegano per selezionare i task. E' definito, a tal proposito, il concetto di team valido.

> *Un **team valido** è un insieme di agenti $\{a_i\}$ ai quali corrisponde un insieme di nodi $\{v_i\}$ che formano un sottografo connesso di G e del quale l'insieme di skill $\{\sigma_i\}$ soddisfa gli skill richiesti per un dato task K.*

3.3.2 Risultati sperimentali

Tramite alcuni esperimenti virtuali sono state analizzate varie topologie di rete e sperimentate varie strategie di allocazione per questo modello di team formation. Matthew E. Gaston (2005) ha dimostrato che la struttura della rete influisce pesantemente sulla performance organizzazionale del team formation. Risulta quindi che è ideale utilizzare strategie di assegnazione che vengano influenzate il meno possibile dalla struttura della rete. Vediamo, ad uso esplicativo, un risultato preliminare che consegue da un esperimento virtuale effettuato da Gaston e desJardins.
I risultati dell'uso di una semplice strategia di selezione di agenti sono mostrati in figura 12. Per questa istanza del modello, sono stati selezionati agenti non impegnati, in maniera casuale. Se un dato task ha almeno un agente già impegnato, altri agenti possono essere contattati (se almeno uno dei loro agenti adiacenti è impegnato su quel task). Essi poi scelgono di auto-assegnarsi ad un task con probabilità proporzionale al numero di skill già soddisfatti per quel task (cioè, un task che ha più agenti impegnati è più probabile che divenga attivo).

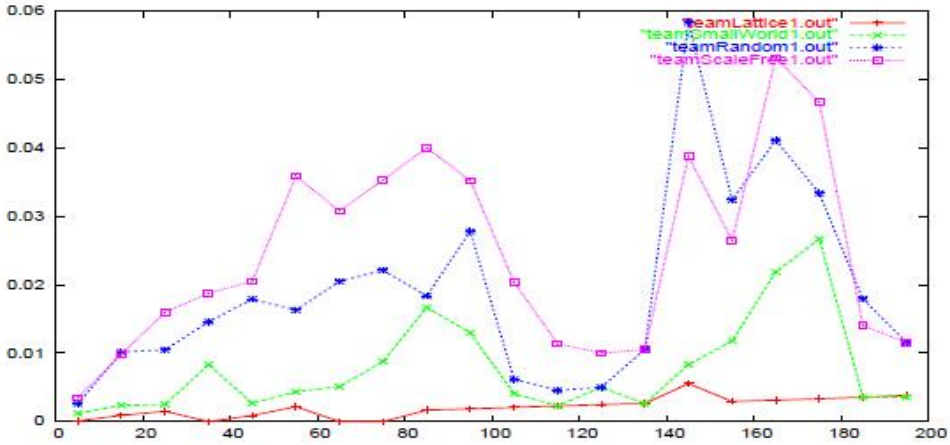

Figura 12 : Efficienza organizzazionale per un'azienda con N = 100, |K| = Σ= 25, e α = 2δ = 4. Il rapporto di introduzione del task, μ, varia lungo l'asse delle ascisse e l'efficienza è mostrata lungo l'asse delle ordinate. Le curve rappresentano alcuni modelli di rete: ad invarianza di scala (- □ -), random (- * -), small-world (-x-), e lattice (- + -).

I risultati di figura 12 mostrano che la struttura di rete influisce pesantemente sulle probabilità che l'organizzazione ha di formare team che soddisfino i task.
L'efficienza è misurata come una proporzione sul numero totale di task che l'organizzazione può soddisfare. La rete a invarianza di scala (*Scale-Free network*) è la più efficiente per questa particolare strategia di selezione di task. Tale risultato è giustificato dall'esistenza di particolari nodi nella rete, fortemente connessi, definiti *hub* (Barabasi, 2002). Questi hub facilitano la formazione dei team e assicurano che un grande numero di skill può essere soddisfatto contemporaneamente.
Basandosi su questi risultati, Gaston e desJardins stabilirono che la struttura di rete influisce pesantemente sulle dinamiche del team formation.

3.4 Ricerca del Team in un Social Network

Theodoros Lappas (2009) è stato il primo a considerare il problema del Team Formation in presenza di una social network di individui. Egli ha effettuato studi su diverse istanze di questo problema, le ha rigorosamente analizzate ed ha presentato gli algoritmi per la sua risoluzione.
In maniera sperimentale egli provò che la sua definizione del problema, cosi come gli algoritmi proposti, risultarono efficienti, dando risultati vantaggiosi ed intuitivi.
Lappas si prefisse di studiare il problema della ricerca di un gruppo di individui capaci di interagire tra loro al fine di creare un team capace di portare a termine uno specifico task.
Assunto che gli *n* candidati siano allocati su di un social graph (come descritto nel prossimo paragrafo), la notazione rivoluzionaria riguardò la particolare introduzione dei pesi sugli archi del grafo. Tali pesi infatti vengono interpretati come segue: un arco con peso basso tra i nodi *i* e *j* implica che i candidati *i* e *j* possono collaborare e/o comunicare più facilmente rispetto ai candidati connessi da un arco con grande peso. Questi pesi possono essere contestualizzati in modi differenti e in diversi domini di applicazione.
Per esempio, in un'azienda, il peso tra due dipendenti potrebbe essere relativo alla distanza del percorso da un dipendente all'altro nell'organigramma aziendale. In una comunità di ricerca scientifica, il peso tra due scienziati è relativo al numero totale di pubblicazioni che li vedono come co-autori. Per calcolare i pesi potrebbero essere usate anche le relazioni interpersonali tra gli individui. Dato un task *T* che richiede un set di skill, il nostro obiettivo è trovare un set di individui $X' \subseteq X$, cosicché ogni skill richiesto in *T* è posseduto da almeno un individuo in X'.
In aggiunta, i membri di X' potrebbero definire un sottografo in G con un più basso costo di comunicazione. Il costo di comunicazione misura come effettivamente i membri del team possano collaborare: più basso è il costo di comunicazione e migliore è la qualità del team.

3.4.1 Notazioni

In questo paragrafo il problema è formalizzato in maniera più precisa. E' considerato un pool di candidati consistenti di *n* individui $X=\{1, \dots, n\}$. E' assunto anche che $A=\{a_1, \dots, a_m\}$ sia un universo di *m* skill. Ogni individuo *i* è associato ad un set di skill $X_i \subseteq A$. Se $a_j \in X_i$ significa che l'individuo *i* ha skill a_j; altrimenti l'individuo *i* non ha skill a_j . Spesso, per riferirsi alla risorsa, è usato anche solo l'insieme di skill che l'individuo possiede. Il sottoinsieme di individui $X' \subseteq X$ possiede skill a_j se esiste almeno un individuo in X' che ha a_j .
Un task T è un semplice sottoinsieme di skill richiesti per eseguire un lavoro. Posto $T \subseteq A$. Se $a_j \in T$ significa che lo skill a_j è richiesto dal task T.
E' possibile anche definire il *cover* di un set di individui X' che rispetti il task T, denotato da C(X',T), come l'insieme di skill richiesti da T e per i quali esiste almeno un individuo X' che li possiede. Cioè, $C(X',T) = T \cap (\bigcup_{i \in X'} X_i)$.

Dato uno skill $a \in A$, il suo *support set* (o semplicemente *support*), denotato da S(*a*), è costituito dall'insieme degli individui in X che hanno questo skill. Cioè, S (a) = {i| $i \in X$ ed $a \in X_i$}.
Come già discusso, per convenienza, è assunto che gli individui siano organizzati in un grafo indiretto e pesato G(*X,E*). Ogni nodo di G corrisponde ad un individuo in *X*. *E* è un insieme di archi connessi ai nodi. Gli archi di G sono pesati; gli archi di basso (o alto) peso rappresentano basso (o alto) costo di comunicazione tra i nodi che essi connettono.
Senza perdita di generalità, è assunto che il grafo G sia connesso, per cui è possibile trasformare ogni sottografo disconnesso con uno connesso semplicemente aggiungendo gli archi con pesi molto alti tra ogni coppia di nodi che appartiene alle differenti componenti connesse. E' possibile scegliere tale peso come la somma di tutti i percorsi corti in G.
Per ogni due nodi *i, i'* ∈X è definita la funzione distanza *d(i,i')* come il peso del più corto percorso tra *i* e *i'*. L'insieme dei nodi lungo lo shortest path tra i e i' è definito da *Path* (*i, i'*). Separatamente dal calcolo della distanza tra i due nodi in G, nasce il bisogno di definire la distanza tra un nodo $i' \in$ X e un insieme di nodi X'⊆ X. Questa particolare funzione di distanza è definita come $d(i,X') = \min_{i' \in X'} d(i,i')$. La funzione *Path* (*i, X'*) è utilizzata per rappresentare l'insieme dei nodi lungo il più corto percorso dal nodo *i* al nodo $j = \arg min_{i' \in X'} d(i,i')$.
Infine, dato un grafo G e X'⊆X, G[X'] è usato per denotare il sottografo di G che contiene solo i nodi in X'.

3.4.2 Il Modello

Verranno analizzate le varie istanze del seguente specifico Problema1: *"Dato un set di n individui X ={1, … , n }, un grafo G (X,E), e task T, troviamo X'⊆X, cosicché C(X',T) = T e il costo di comunicazione Cc(X') sia minimo".*
Per evidenziare le generalità più integrali del problema originale del Team Formation, si è deliberatamente evitato di definire il costo di comunicazione nella definizione del Problema1. Ci si focalizza infatti su 2 istanze del problema basandosi su altrettante definizioni della funzione *communication-cost*. Queste istanze sono state scelte per motivi di praticità, semplicità ed intuitività.
(1) Diametro (R): Dato un grafo G(X,E) e un set di individui X'⊆X, il *diameter communication cost* di X', denotato da Cc-R(X') è definito come il diametro del sottografo G[X']. Ricordiamo che il diametro di un grafo è il più corto percorso tra i due più distanti nodi in un grafo.
(2) Mimum Spanning Tree (MST): Dato un grafo G (X,E) e X'⊆X il *MST communication cost* di X', denotato con Cc-MST(X') è definito come come il costo del minimum spanning tree sul sottografo G[X']. Ricordiamo che il costo di uno spanning tree è semplicemente la somma dei pesi dei suoi archi.
Il problema Team Formation con funzione di comunicazione Cc-R, è detto problema DIAMETER-TF. Similarmente MST-TF problem sarà il problema di Team Formation con la funzione di comunicazione Cc-MST.
E' dimostrabile che il *DIAMETER-TF problem è NP-completo,* e che il problema Diameter-TF è NP-hard.
Per il MST-TF problem, abbiamo il seguente robusto risultato: "*Il MST-TF problem è NP-completo."*
Nella definizione del problema e delle sue specifiche, Lappas si è concentrato sulla minimizzazione dei costi di comunicazione tra i membri del team. Altre nozioni dell' "efficienza" di un team possono portare ad una differente funzione di ottimizzazione. Per esempio, se si tralasciasse il costo di comunicazione, si potrebbe definire come obiettivo quello di trovare X'⊆X, tale che *C*(X',T)=T e |X'| siano minimi. In tal modo la definizione del problema ignora l'esistenza del grafo G(X,E), e diventa un istanza del classico problema Set Cover, che può essere risolto con l'algoritmo standard GreedyCover.

Ottimizzando sia la cardinalità del team che il costo di comunicazione tra i suoi membri, è possibile richiedere la minimizzazione di una funzione della forma $\alpha \cdot |X'| + (1-\alpha) \cdot Cc(X',G)$, dove $\alpha \in [0,1]$. Per $\alpha=1$ il problema ricerca i team con cardinalità minima. Per $\alpha=0$ il problema diventa il Team Formation Problem. Questa ipotesi è valutata nel paragrafo 3.6 (*Il Modello on line*).
Comunque per valori di α in (0,1) non è chiaro se ha senso ottimizzare questa funzione alternativa; questo soprattutto perché i due termini nella somma sono in scala differente e non si conosce come queste due metriche possano relazionarsi.
Alternativamente, se si definisce il problema come un'ottimizzazione bi-obiettivo, la grandezza del team e il costo di comunicazione potrebbero essere presi in considerazione simultaneamente. In tali casi l'obbiettivo è trovare le soluzioni ottime di *Pareto*. Per molti problemi, il set di soluzioni ottime di Pareto è esponenziale rispetto alla grandezza dell'input cosicché tali soluzioni non possono essere trovate in tempo polinomiale. In genere, più grandi sono i team e più alto risulta il costo di comunicazione.
Nel nostro caso di studio, è assunto che entrambi gli individui o nessuno di essi posseggano un dato skill e non è considerata una specifica gradazione delle abilità dei nodi.
Similmente per i task; è assunto che un task richieda un certo insieme di skill, senza tener conto della particolare importanza che i singoli skill possano avere per il completamento del task. Detto ciò, una onesta generalizzazione del problema del Team Formation potrebbe essere la sua variante graduata. In una simile variante, il grado delle abilità di un individuo e l'estensione allo skill richiesto per il completamento di un task, possono essere modellate sul valore di un peso intero, per es. $\{0,1, \ldots, \delta\}$. In questo caso, le specifiche del task, per ogni skill richiesto $a_j \in T$, indicano il livello minimo richiesto δ_j.
Similmente, per ogni individuo i con skill a_j, è specificato il livello di questa competenza relativamente ad a_j.
Poi, tutti gli individui con livello di competenza pari al minimo livello richiesto sono capaci di contribuire alla copertura di questo skill per un dato task.
Concettualmente assumiamo che un individuo ha uno skill, solo se il rispettivo livello di competenza è maggiore o uguale di quello richiesto.
In tal modo, questa versione "graduata" del problema diviene identica alla versione base del problema Team Formation, studiato principalmente in questo lavoro.

3.4.3 Gli Algoritmi

Ora presentiamo gli algoritmi per i problemi DAMETER-TF e MST-TF ideati da Lappas-Liu-Terzi. Le soluzioni algoritmiche sfruttano le relazioni tra questi due problemi e rispettivamente i problemi MCC e GST.

- Algoritmi per il problema DIAMETER-TF

L'algoritmo1 mostra lo pseudo-codice dell'algoritmo RarestFirst per il problema Diameter-TF.
Per prima cosa, per ogni skill a richiesto dal task T, è calcolato S(a), come support di a. Poi, l'algoritmo seleziona lo skill $a_{rare} \in T$ con il support $S(a_{rare})$ di più bassa cardinalità. E' da notare che bisogna includere nella soluzione almeno un individuo dal set $S(a_{rare})$. Tra tutti i candidati del set $S(a_{rare})$, l'algoritmo ne seleziona uno che conduce al più piccolo diametro del sottografo, quando l'individuo più prossimo, in tutti gli altri gruppi supporto S(a) (con $a \in T$ e $a \neq a_{rare}$), è connesso ad esso.

Algoritmo 1. Algoritmo **RareFirst** per il problema Diameter-Tf

Input: Grafo G(*X*,E); vettori degli skill $\{X_1, \dots, X_n\}$; task T

Output: Team $X' \subseteq X$ e il sottografo G[X']

1: ∀ a ∈ T do
2: S(a) = {i | a ∈ X_i}
3: $a_{rare} \leftarrow \arg min_{a \in T} |S(a)|$
4: ∀ i ∈ S(a_{rare}) do
5: ∀ a ∈ T and a ≠ a_{rare} do
6: $R_{ia} \leftarrow d(i, S(a))$
7: $R_i \leftarrow max_a R_{ia}$
8: i* ← arg *min* R_i
9: $X' = i^* \cup \{Path\ (i^*, S(a)) \mid a \in T\}$

In riferimento alla linea 6 dell'Algoritmo1, *d(i,S(*a*))* è semplicemente il *min i'*∈S(a) . Il *Path* (i*, S(a)) (linea 9) è l'insieme dei nodi nel grafo che si trovano lungo il più breve percorso da *i** a *i',* dove *i'* è tal che i'∈S(a) e d(i, S(a)) = d(i*, i').
E' assunto che tutte le coppie contraddistinte dal cammino più corto siano state pre-calcolate, inoltre per il salvataggio degli individui che possiedono uno specifico attributo sono usate le tabelle hash.
Il tempo di esecuzione dell'algoritmo RarestFirst è O(|S(a_{rare})| × *n*). L'analisi del caso peggiore suggerisce che |S(a_{rare})| = O(n). Quindi il tempo di esecuzione del caso peggiore del RarestFirst è $O(n^2)$.
In pratica, il tempo di esecuzione dell'algoritmo è molto più piccolo di quello del caso peggiore.
Visto che la distanza calcolata *d* è una metrica, è possibile affermare la seguente proposizione come fattore di approssimazione per l'algoritmo RarestFirst:
"Per ogni funzione graph-distance d che soddisfa la disuguaglianza triangolare, il costo CC-R della soluzione X', dato dal RarestFirst per un task dato, è al più il doppio del costo CC-R della soluzione ottima X, Cioè, CC-R(X') ≤ 2 * CC-R(X*)."*

- Algoritmi per il problema MST-TF

In questa sezione saranno descritti 2 algoritmi per risolvere il problema MST-TF: gli algoritmi *CoverSteiner* e *EnhancedSteiner* (Lappas, 2009). Entrambi gli algoritmi sono motivati dalla somiglianza del MST-TF ai problemi dello Steiner tree.
Il CoverSteiner algorithm:

Algoritmo 2. Algoritmo **CoverSteiner** per il problema MST-TF

Input: Grafo G(*X*,E); vettori degli skill $\{X_1, \dots, X_n\}$; task T

Output: Team $X' \subseteq X$ e il sottografo G[X']

1: $X_0 \leftarrow$ GreedyCover (*X*, T)
2: $X' \leftarrow$ SteinerTree (G, X_0)

La prima Euristica presentata per il problema MST-TF procede in due step.
Nel primo, la social network è ignorata e l'algoritmo si focalizza sulla ricerca di un insieme di individui $X_0 \subseteq X$ tali che $\cup_{i \in X_0} X_i \supseteq T$.
Nel secondo step, l'algoritmo trova l'albero dei costi minimi che attraversa tutti i nodi in X_0, e possibilmente altri nodi in $X \backslash X_0$. In questo modo è restituito un insieme di nodi X' tali che $X_0 \subseteq X' \subseteq X$.

Questo algoritmo a due step è detto algoritmo CoverSteiner e lo pseudo-codice è mostrato dall'Algoritmo2.
L'obiettivo del primo step è quello di risolvere un istanza del classico problema del Set Cover: l'universo di elementi che deve essere coperto è costituito dai requisiti del task T e ogni individuo in X è un sottoinsieme dell'universo. Per giungere alla risoluzione di tale problema è utilizzato l'algoritmo standard GreedyCover per il problema del Set Cover. L'algoritmo GreedyCover che aggiunge ad ogni step t l'individuo X_t che possiede al più i requisiti già richiesti in T, è una procedura greedy iterativa.
Nel suo secondo step, l'algoritmo CoverSteiner risolve un istanza del problema dello Steiner Tree sul Grafo G.
E' utile rammentare, a tal punto, che sul problema standard Steiner Tree, è stato determinato un grafo non diretto con costi sugli archi non negativi.
I vertici di questo grafo sono partizionati in due insiemi: i vertici-requisiti e i vertici Steiner.
Il problema Steiner Tree poi, per il minimum-cost tree, chiede in input il grafo che contiene tutti i vertici richiesti e ogni sottoinsieme dei vertici Steiner.
Nel caso d'interesse, l'insieme dei nodi X_0 riportato dall'algoritmo GreedyCover corrisponde all'insieme dei vertici richiesti, mentre i vertici in $X \backslash X_0$ rappresentano i vertici Steiner.
Dato il grafo G(X,E), l'obiettivo della linea 2 dell'Algoritmo2 è di trovare la soluzione X' che minimizza CC-MST (X'), sotto il vincolo che $X' \supseteq X_0$.
Esistono molti algoritmi per la risoluzione del problema classico dello Steiner Tree. Lo pseudocodice dell'algoritmo utilizzato per gli esperimenti è illustrato qui di seguito.

Algoritmo 3. Algoritmo **SteinerTree**.

Input: Grafo G(*X*,E); nodi richiesti X_0 e nodi Steiner X/X_0
Output: Team $X_0 \subseteq X' \subseteq X$ e il sottografo G[*X'*]

1: X'← v con v nodo casuale di X_0
2: while $(X_0 \backslash X') \neq 0$ do
3: $v^* \leftarrow arg \arg min_{u \in X_0 \backslash X'} \, d(u,X')$
4: if $Path\ (v^*, X') \neq 0$ then
5: $X' \leftarrow X' \cup \{\ Path\ (v^*, X')\}$;
6: else
7: Return failure

L'algoritmo 3 (in realtà un'euristica greedy per lo Steiner Tree) aggiunge, incrementalmente, alla soluzione corrente X' i nodi dal set X_0 dei requisiti.
Ad ogni step viene aggiunto un singolo nodo da X_0; questo è il nodo *v** che ha la minima distanza dall'insieme di nodi X' già aggiunti alla soluzione (linea 3).
Se esiste *v**, verrà aggiunto, con tutti i nodi nel più corto percorso da esso verso X', all'insieme delle soluzioni. Altrimenti, viene restituito *failure*.
Il tempo di esecuzione dell'algoritmo CoverSteiner è la somma dei tempi di esecuzione del GreedyCover più lo SteinerTree.
Il tempo richiesto per l'esecuzione dell'algoritmo GreedyCover è O(|T| x |X|) oppure O(*mn*).
Il tempo richiesto per l'esecuzione dello SteinerTree mostrato nell'Algoritmo3 è $O(|X_0| \times |E|)$.
Così, nel caso peggiore, il tempo di esecuzione del CoverSteiner è $O(n^3)$ ($|X_0| = O(n)$ e $|E| = O(n^2)$).
Nella realtà comunque le cardinalità degli insiemi X_0 ed E sono spesso molto minori rispetto ai loro limiti, relativi al caso peggiore.
Il principale svantaggio dell'algoritmo CoverSteiner è che, nel primo step, esso ignora completamente la struttura del grafo evidenziato. Questo può portare a team con alti costi di

comunicazione, oppure potrebbe anche portare al *failure*, anche nei casi dove la soluzione al problema MST-TF esiste.
L'inadeguatezza dell'algoritmo CoverStainer può essere attenuata dall'algoritmo EnhanceSteiner. Esso effettua lo start dal primo grafo G aumentato con i nodi aggiuntivi e gli archi per formare l'ehanced graph H. Poi, lo SteinerTree è richiamato per risolvere il problema dello STEINER TREE sul grafo aumentato H.

Algoritmo 4. Algoritmo EnhancedSteiner per il problema MST-TF.

Input: Grafo G(*X*,E); vettori degli skill $\{X_1, \dots, X_n\}$; task T.
Output: Team $X' \subseteq X$ e il sottografo G[X'].
1: H← EnhancedGraph (G,T)
2: $X_H \leftarrow$ SteinerTree (H, $\{Y_1, \dots, Y_n\}$)
3: $X' \leftarrow X_H \setminus \{Y_1, \dots, Y_n\}$

Il task T è eseguito secondo i *k* skill richiesti, cioè T = $\{a_1, \dots, a_k\}$. La routine Enhance (linea 1) effettua un passaggio lineare sul grafo G e lo incrementa come segue: un nodo addizionale Y_j è creato per ogni skill $a_j \epsilon$ T. Cosicché un nuovo vertice Y_j è connesso ad un nodo $i \in$X se e solo se $a_j \in$X.
La distanza tra il nodo Y_j e i nodi $i \in S(a_j)$ è fissata come $d(Y_j,i)$ = D dove D è un grosso numero reale, più grande della somma della distanza tra tutte le coppie di nodi nel grafo G.
In fine, ogni nodo $i \in$X che ha abilità X_i è sostituito da un esclusiva *cricca* (Cap 3.1.3) C_i di taglia $|X_i|$.
Ogni nodo del grafo completo C_i potrebbe essere considerato come una copia dell'individuo i che ha solo un singolo, distinto skill dell'insieme X_i.
La distanza tra ogni due nodi nella cricca C_i è settata a zero. Ogni nodo di C_i mantiene tutte le connessioni esistenti del nodo *i* al resto del grafo – includendo le connessioni ai nodi $\{ Y_i, \dots, Y_k\}$.
I nodi dell'insieme X_H, che partecipa allo Steiner tree del grafo aumentato H, sono trovati dalla chiamata all'algoritmo Steiner Tree con i nodi richiesti $Y_1, \dots, Y_k$.
Nello step finale, l'algoritmo rimuove dal set X_H i nodi aggiunti artificialmente $Y_1, \dots, Y_k$ (e i loro archi incidenti) per ottenere la soluzione definitiva X'. La seguente definizione nasce dal rispetto di questo algoritmo.
Siano rispettivamente X_H^* l'insieme dei nodi nello Steiner tree ottimale del grafo aumentato H, e X* il team ottimale per il problema MST-TF. Sia Cc-Mst (X*) = Cc-Mst($X_H^* \setminus \{Y_i, \dots, T_k\}$). Cioè, se i nodi $Y_1, \dots, Y_k$ (e i loro archi incidenti) sono rimossi dalla soluzione ottima del problema dello Steiner tree sul grafo aumentato H, allora i nodi rimanenti formano la soluzione ottima al problema Mst-TF.
Questa sostituzione di ogni individuo *i* con una cricca C_i di taglia $|X_i|$ è solo concettuale; in pratica, l'implementazione dell'algoritmo non richiede questo. Il grafo aumentato H conterrà solo k più nodi rispetto al grafo in input G, vale a dire i nodi $Y_1, \dots, Y_k$.
Perciò seguendo l'analisi dello SteinerTree fatta nella sezione precedente, si otterrà che il tempo di esecuzione dell'algoritmo Enhanced-Stainer è O(k × |E|).
L'algoritmo Enhanced-Stainer è infatti motivato dalla evidente similarità tra il problema Mst-Tf e il problema Group Steiner Tree (GST). In generale, al posto dell'algoritmo EnhancedSteiner per risolvere il problema MST-TF, può essere usato ogni altro algoritmo (di approssimazione) per il problema GST.
Il miglior rapporto di approssimazione raggiunto dall' algoritmo è O($\log^3 n$ logk).

3.5 L'approssimazione greedy

Partendo dall'assioma che la performance di un individuo in un gruppo dipende, non solo dai suoi skill, ma anche dalle sue relazioni con gli altri membri, può essere possibile utilizzare diverse strategie, tenendo presente le sinergie che i social network mettono a disposizione. Sarà analizzato il team-formation nel contesto di una grande organizzazione che vuole formare team multipli composti dai suoi collaboratori. Alcune organizzazioni possono disporre di competenze che vanno, ad esempio, dai servizi di intelligence con molti analisti a quelli di consulenza con molti consulenti, tutti hanno differenti esperienze ma soprattutto i membri hanno diversi tipi di skill e di personalità. L'organizzazione deve suddividere i suoi membri in team, completando ogni team con le specifiche liste di profili, ad ognuno dei quali è associato anche una differente retribuzione. Gli skill di ogni membro vengono catalogati in diversi livelli, a seconda della probabilità di completare con successo un particolare tipo di task.
Connettere un membro ad ogni altro, come avviene tipicamente nei social networks, garantisce essenzialmente un grande aspetto positivo: essi possono aiutarsi vicendevolmente al di fuori del task d'interesse, innalzando le probabilità di successo dell'intero progetto. L'azienda ha quindi un insieme di progetti da realizzare ed ogni progetto consiste di un insieme di task di differenti tipi. Ogni task è caratterizzato dal suo valore, il suo tipo, e il progetto di cui fa parte. L'idea alla base è che l'organizzazione è in realtà un gruppo di esperti. Ogni esperto ha esperienza in un particolare tipo di task a prescindere dal progetto a cui questo task appartiene. La condivisione fruttuosa dell'informazione su un particolare task può avere effetti sinergistici sulle performance relative ad altri task.
E' possibile ora introdurre un modello che sintetizzi gli elementi prima descritti, e poi dimostrare che, prendendo in considerazione la struttura di un social network, è possibile avere significativi benefici in termini di ottimalità complessiva dell'allocazione di task. Sarà introdotto poi un algoritmo greedy per tale problematica, considerando la struttura di una rete sociale, e sarà quindi dimostrato in maniera sperimentale che esso fornisce una buona approssimazione per l'allocazione ottima su piccoli grafi. Mentre, saranno raggiunti significativi benefici su grandi grafi solo se è considerata la soluzione ottima che non contempla la struttura di un social network.
In fine, tale algoritmo greedy è usato per esplorare le proprietà di differenti tipi di social network; scopriremo che i grafi che danno riscontro positivo sono costituiti da piccole reti, da random network, mentre gli ultimi in tale ordine sono le reti connesse secondo preferenze date. Il modello qui di seguito illustrato, è quello che logicamente più si avvicina a quello di Lappas (visto nel paragrafo precedente), che considera anche la struttura del grafo sociale degli individui durante la formazione dei team. Lo scopo di Lappas è però differente: nel suo modello, gli agenti hanno skill binari, e ogni task è formato da appropriati insiemi di skill. Questa sezione invece si focalizza sull'allocazione ottima di risorse in un framework teorico ben più complesso.

3.5.1 Il modello matematico

Ci sono *n* esperti $E_1, \dots, E_n$ che lavorano per un'organizzazione. Il social network che rappresenta le relazioni sinergistiche è dato da S. Due esperti sono connessi in S se accrescono la performance di ogni altro che lavora nello stesso team assegnato ad un particolare progetto.
Ci sono *z* differenti tipi t_i di task. Ogni esperto ha un differente livello di skill associato ad ognuno di questi *z* tipi di task. Gli skill di un esperto E_i sono così rappresentati da un vettore $S_i = (s_{i1}, s_{i2}, \dots s_{iz})$ dove s_{ij} è la probabilità che E_i può completare con successo un task di tipo j. Questa è una misura generale; essa può includere non solo gli skill tecnici ma anche i profili interrelazionali

oppure organizzazionali. E' assunto che il manager responsabile per l'allocazione di un task (M) conosca il vettore di skill per ogni esperto.
Il tempo scorre in step discreti e ad ogni unità di tempo *T*, i manager *M* assegnano il lavoro agli esperti per la prossima unità *T*; questo periodo di unità T sarà chiamato "*round*".
Il manger *M* alloca *m* differenti progetti $R_1,\ldots,R_m$. Ogni progetto R_i ha q task $T_1^1,\ldots,T_q^i$. La distribuzione dei tipi di task è la stessa per ogni progetto. Ogni task T_i del progetto R_j (rappresentato da T_i^j) è associato al valore V_{ij} che descrive i guadagni ricevuti dall'organizzazione per il completamento con successo del task (questa è una misura diretta di utilità). Ogni task disponibile nell'organizzazione è caratterizzato da tre attributi: tipo di task, progetto a cui appartiene e valore.
E' costruito un vettore di probabilità di completamento con successo di task in un progetto dato R_k per ogni esperto E_i, P_i^k: $(p_{i1}^k, \ldots, p_{iq}^k)$ da S_i, che è il vettore degli skill, definito prima.
E' assunto che il manager ri-allochi ognuno dei *q* task differenti in tutti e *m* i progetti all'inizio di ogni round; perciò in tutto ci sono m_q task da allocare tra gli *n* esperti. I valori dei task non restano gli stessi ad ogni round, cosicché l'algoritmo di allocazione deve essere eseguito all'inizio di ogni round. I task sono assegnati in modo tale che essi possano essere terminati in un round. Ad un esperto può essere assegnato un solo task.
Tale modello prevede che il social network *S* di esperti sia conosciuto dal manager. Questa assunzione non è realistica, perché il manager acquisisce questa informazione in maniera graduale, man mano che lavora con questi esperti. Per ovviare a tale problematica, esistono vari test, come (Kolbe) o (Briggs), grazie ai quali i manager possono effettuare una prima selezione, trovando particolari tipologie di esperti, utili per costruire una rete basata sulle compatibilità tra i differenti tipi di agenti.
Verrà definito ora un modello che esprime come gli utenti possano aiutarsi reciprocamente nell'eseguire i loro task.
Sia *W* la matrice che rappresenta le "amicizie" adiacenti create dal social grafo *S*:

$$w_{ij}=\begin{cases}1 \text{ se } E_i \text{ e } E_j \text{ sono "amici",}\\ 0 \text{ altrimenti.}\end{cases}$$

f_i^k rappresenta il numero di amici di E_i tra gli esperti assegnati ad un task di un progetto R_k. La rete è modellata secondo l'incremento di probabilità che un esperto completi con successo un task. $B = 1 - e^{-f_i^k}$ è un coefficiente che rappresenta il miglioramento della performance di un esperto, risultante dalla collaborazione con gli amici su di un task; *B* definisce la frazione del gap di performance ($1-p_{ij}^k$) che è coperto dalla collaborazione. Dalla definizione, $0<B<1$.
La probabilità di aumentare la percentuale di realizzazione con successo del task T_j nel progetto R_k, denotata con p_{ij}^{k+} è definita come:

$$p_{ij}^{k+} = \frac{p_{ij}^{k+}}{1-B(1-\max(p_{ij}^k,1/2))}$$

Dove più è alto il coefficiente B, maggiore è la miglioria. L'accrescimento è naturalmente limitato dal fattore $1-p_{ij}^k$.
Il denominatore dell'espressione di probabilità è al più $min(p_{ij}^k, 1/2)$, quindi la probabilità di miglioramento è minore di $min(2p_{ij}^k, 1)$.

3.5.2 L'inflluenza della struttura

L'algoritmo per l'allocazione dei task, secondo il *modello greedy*, prevede ancora qualche definizione.

L'indicatore a_{ij}^k è definito come segue:

$$a_{ij}^k = \begin{cases} 1 \text{ se } E_i \text{ è allocata sul task } T_j \text{ nel progetto} \\ R_k, \\ 0 \text{ altrimenti.} \end{cases}$$

Ammesso che non ci siano effetti sulla rete, la performance di ogni task dipende solo dai livelli di skill degli esperti selezionati. Quindi, l'utilità totale attesa è data da:

$$U = \sum_{i=1}^{n} \sum_{j=1}^{q} \sum_{k=1}^{m} a_{ij}^k p_{ij}^k v_j^k \qquad (3.3)$$

L'obiettivo diventa trovare le variabili di allocazione a_{ij}^k come massimizzato dall'equazione (3.3).

Dalle conoscenze del manager sui livelli di skill di tutti gli esperti, questo problema riduce al massimo il costo del matching in un grafo bipartito con gli esperti da un lato e i task dall'altro, come mostrato in figura 13.

Il peso dell'arco tra l'esperto E_i e il task T_{jk} rappresenta il valore atteso se E_i è assegnato al task T_j^k, ossia è uguale al prodotto tra la probabilità del completamento con successo p_{ij}^ke il valore associato con quel task V_j^k. Negli esperimenti è stato utilizzato un algoritmo noto come "algoritmo ungherese" (Lovász, 1986) per risolvere il problema del matching e trovare l'allocazione ottimale, massimizzando i pesi.

Anche se il manager volutamente ignori l'efficacia della rete, gli effetti in termini di performance complessiva sono comunque evidenti. Perciò, quando le soluzioni che tengono conto di tali effetti della rete vengono comparate, si includono le "reti nascoste" le quali sono definite man mano che i manager acquisiscono informazioni nel processo di allocazione.

Se invece si considera la struttura della rete, l'utilità attesa totale può essere calcolata con l'aggiornamento della probabilità di successo nell'equazione (3.3):

$$U_f = \sum_{i=1}^{n} \sum_{j=1}^{q} \sum_{k=1}^{m} a_{ij}^k p_{ij}^{k+} v_j^k \qquad (3.4)$$

C'è bisogno di assegnare a_{ij}^k in modo che l'utilità totale attesa U_f sia massimizzata.

Opt = $max_{a_{ij}^k}(U_f)$ rappresenta la massima utilità attesa che può certamente essere raggiunta. Visto che questo problema è computazionalmente difficile da risolvere in maniera ottimale, sembra opportuno concentrarsi sull'algoritmo di approssimazione greedy.

3.5.3 Un'approssimazione Greedy

E' proposto un semplice algoritmo di approssimazione Greedy che può essere usato per allocare task come segue.

Il primo step prevede la costruzione di un grafo pesato bipartito con gli esperti da un lato e i task dall'altro (separatamente dal grafo che rappresenterà i legami sociali tra gli esperti). Come rappresentato in figura 13, il peso su un arco tra esperti E_i e task T_{jk} è uguale al valore di ritorno atteso se l'esperto E_i è assegnato al task T_{kj} cosicché il valore di ritorno associato al task sia T_{jk}.

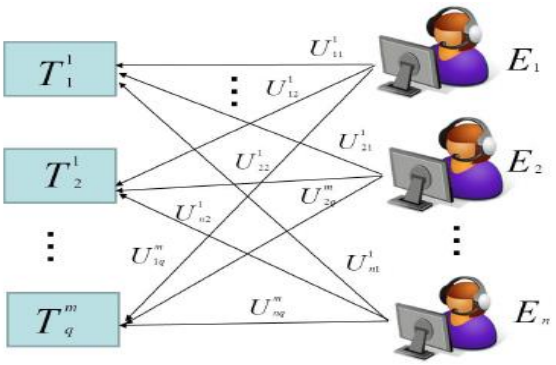

Figura 13 : Il lato sinistro rappresenta i task e il destro gli esperti. Il peso U_{ij}^k sull'arco tra gli esperti E_i e il task T_j^k è il valore di ritorno atteso se E_i è assegnato al task T_j^k. L'obiettivo è di trovare il matching tale che il valore di ritorno atteso sia massimizzato. E' facile osservare che l'allocazione ottima è la stessa del matching tra i pesi massimi.

Nel secondo step viene selezionato il link con il massimo peso e al task corrispondente è assegnato l'esperto con maggiori probabilità relative di successo.
Vengono poi aggiornate le probabilità di successo di tutti gli esperti connessi al task in questione nel social graph relativamente al progetto k. Questo processo è ripetuto fino a che non ci sono più task o non ci sono altri esperti. La complessità totale di questo algoritmo è $O(\min(n,mq)nmq)$.

3.5.4 Valutazioni sperimentali

Per quanto riguarda i risultati sperimentali, sono stati considerati tre modelli di rete:

1. Random Graph Network (RGN): E' usato il modello G(n,p) Erdos-Renyi (Erdős, 1959) per la generazione del grafo random (Cesa-Bianchi, 2009).
2. Small World Network (SWN): E' usato il β model di *Watts* e *Strogatz* (1998). La rete è rappresentata da una tupla (n,k,β), dove *n* è il numero di nodi nella rete, *k* è il grado di ogni nodo, e *β* è un parametro che rappresenta la casualità, tale che $0 \leq \beta \leq 1$.
3. Preferential Attachment Network (PAN): Questo modello di rete implementa l'assioma "rich get richer". Per generare queste reti è usato il meccanismo di *Barbasi* (1999).

Test dell'Algoritmo Greedy. La performance dell'algoritmo greedy è comparata all'allocazione ottima. Come prova di validazione, Chhabra considerò i grafi con pochi esperti, tale che fosse possibile una ricerca di forza-bruta per trovare la soluzione ottima. Ad oggi non si conoscono risultati efficaci circa la computazione dell'allocazione ottima quando si prende in considerazione la struttura della rete.
Ad esempio è stata considerata una rete con 10 esperti e 2 progetti, ognuno con 5 task e grado medio pari a 2. Inoltre è assunto che tutti gli esperti siano equamente competenti nell'eseguire ogni task. La probabilità di completare con successo un task per ogni esperto è 0.2 . Il valore di ritorno di ogni task è un campione di una distribuzione Gaussiana con $\mu = 1$ e $\sigma = 0.05$.

Network Type	(k)	β	Pr(Success)	Utility w/o network	Opt Utility	Greedy
SWN	2	0.25	0.2	2.75	3.30	3.20
SWN	2	0.10	0.2	2.76	3.30	3.24
SWN	2	0.00	0.2	2.77	3.29	3.29
PAN	2	-	0.2	2.64	3.06	2.72
RGN	2	-	0.2	2.66	3.11	2.98
RGN	2	-	$\mathcal{N}(0.2, 0.05^2)$	2.70	3.16	3.04

Tabella 4 - Paragone tra allocazione greedy e allocazione ottima. k denota il grado medio. 'Utility w/o network' è l'utilità delle allocazioni senza considerare il social network, cioè usando il massimo peso secondo l'approccio descritto nella sezione precedente, ma l'utilità è calcolata usando l'equazione (3.4); 'Opt Utility' è calcolata trovando l'allocazione che massimizza l'equazione (3.4); Greedy alloca i task usando l'algoritmo prima descritto (l'utilità è calcolata ancora utilizzando l'equazione (3.4)). Due chiavi di osservazione sono che: (1) Non considerando la struttura della rete l'allocazione è significativamente sub ottimale; (2) L'allocazione greedy rende quasi ottima la performance.

I risultati riportati nella tabella 4 mostrano che l'approssimazione greedy genera un risultato vicino all'ottimo.
E' interessante notare la variazione nella performance rispetto alla topologia della rete. Alla modifica della struttura della rete corrisponde un sostanziale mutamento del fattore di utilità: ad esempio, l'ordine di utilità ottima risulta essere SWN>RGN>PAN. Ciò accade anche se il grado medio è mantenuto costante attraverso i diversi tipi di rete. Per piccole reti (SWN), non appena la probabilità di rifacimento β decresce, mostrano un piccolo incremento nella performance dell'algoritmo greedy. Ciò è vero però solo per piccole reti, infatti, l'approssimazione può essere peggiore nelle grandi reti. Le proprietà di approssimazione di questo algoritmo costituiscono un interessante argomento di ricerca tuttora aperta.
Questo esperimento dimostra anche che, durante il team formation, c'è un significativo vantaggio nel considerare gli effetti della rete. Assodato che non è possibile calcolare l'utilità ottima per grandi reti a causa dei costi computazionali, se si potesse dimostrare che l'utilità ottenibile utilizzando l'algoritmo greedy sia significantemente più alta rispetto a quando gli effetti della rete non sono presi in considerazione, si otterrebbe il limite inferiore sul guadagno realizzabile. Di seguito sarà analizzato il differenziale di utilità tra l'allocazione greedy e l'allocazione ottima, ignorando nella fattispecie, gli effetti del network su grandi reti.

3.5.5 Allocazione greedy vs allocazione ottima

Real World Networks: Il termine Utility Ratio (UR) è definito come il rapporto tra l'utilità raggiunta usando l'algoritmo greedy e l'utilità raggiunta dal processo di ottimizzazione dell'allocazione senza considerare gli effetti sulla rete (sebbene siano sempre presi in considerazione gli effetti sulla rete nel calcolo dell'utilità reale).
La tabella 5 mostra che il guadagno possibile dalle strategie di allocazione dei task più piccoli diviene realmente molto evidente se applicato a grandi organizzazioni. D'altra parte è possibile che mentre la percentuale di crescita in rapporto alla struttura di rete nell'allocazione è grosso modo equivalente per i tre tipi di rete, l'utilità complessiva tende ad essere più alta soprattutto per piccole reti.

Network Type	(k)	β(Randomness)	Pr(Success)	Utility w/o network	Greedy	UR
SWN	96	0.25	0.2	154.6	192.0	1.24
SWN	96	0.00	0.2	155.0	191.9	1.23
PAN	96	-	0.2	152.0	185.8	1.22
RGN	96	-	0.2	155.1	189.8	1.22

Tabella 5 - Per organizzazioni di taglia media: i risultati sperimentali quando 480 esperti sono assegnati a 48 progetti, ognuno con 10 task. "UR" rappresenta il rapporto di utilità.

Effetti della Taglia del Team e della Connettività: la Figura 14 mostra i benefici raggiunti considerando gli effetti della rete come funzione della taglia del team e della connettività per reti di grafi random. I benefici si riducono con l'incremento della taglia del team se la connettività del social network resta costante. Questo perché, in team di grandi dimensioni, la probabilità di avere assegnato un esperto socialmente cooperativo, anche con allocazione casuale, è alta, rispetto a team di piccola taglia. Inoltre la crescita della connettività comporta una decrescita della presenza di esperti socialmente cooperativi che lavorino allo stesso progetto. Questa diminuzione, dovuta all'incremento di connettività, non è un fattore esplicito al momento dell'allocazione. Sperimentando anche altre tipologie di reti le osservazioni sono le medesime.

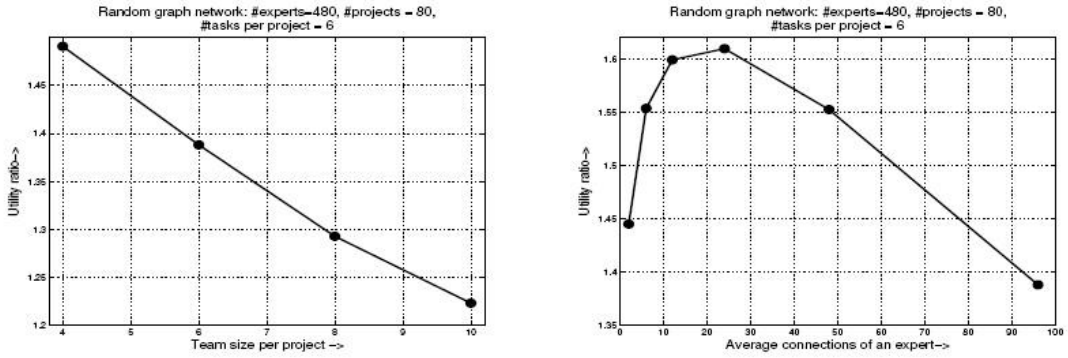

Figura 14: Così come aumenta la taglia del team, aumenta anche la probabilità che un esperto abbia esperti vicini nel suo team, in ogni allocazione; la percentuale di utilità perciò diminuisce con l'incremento della taglia dei team. Inizialmente la percentuale di utilità cresce perché ci sono più esperti disponibili che possono accrescere la produttività degli esperti dati; successivamente però essa diminuisce. D'altronde la probabilità di avere casualmente vicini di esperti che lavorano allo stesso progetto, cresce anche in qualsiasi altra metodologia di allocazione.

E' possibile concludere asserendo che il peso della struttura del social network nell'allocazione di task su network di esperti è certo ed è dimostrabile. Prendere esplicitamente in considerazione la struttura del social network nella determinazione dell'allocazione di esperti sui task, può risultare un elemento particolarmente prezioso per le organizzazioni interessate al team formation.

3.6 Il modello on-line

In questa sezione si presuppone che gli utenti siano "compatibili alla collaborazione". E' possibile infatti considerare uno scenario in cui le persone possiedano differenti skill e la compatibilità tra i potenziali membri del team sia modellata da un social network collaborativo. Una sequenza di task giunge on-line e ogni task necessita di uno specifico set di skill. L'obiettivo è quello di formare un team finalizzato all'esecuzione di ogni task, tale che possegga tutti gli skill richiesti dal task. Inoltre, ad ogni team è richiesto un basso costo di comunicazione e contemporaneamente vogliamo che il carico di lavoro tra le persone sia bilanciato nella maniera più equa possibile. I costi di comunicazione sono dipendenti dalle "connessioni" all'interno della rete; mentre una scelta errata dei task sulle risorse potrebbe generare un carico di lavoro eccessivo per alcune e insufficiente per altre.
Saranno proposti degli algoritmi efficienti che rispettano i vincoli succitati (i team che soddisfano sempre gli skill richiesti, forniscono garanzie di approssimazione rispetto ai costi di comunicazione e sono competitivi rispetto al bilanciamento del carico di elaborazione).

3.6.1 Definizioni

Task. J = { J^j; j = 1, 2,…, k} denota l'insieme dei task (o jobs). Considerato il setting online, il j-simo task arriva al tempo j-simo ed è assegnato al team di esperti prima che arrivi il (j+1)-simo task.

Skill. Ogni task, per essere completato, richiede un sottoinsieme di *m* skill disponibili. E' usata la nozione di skill space S = $\{0,1\}^m$, che è lo spazio che contiene tutti i modi possibili con cui gli m skill possono essere combinati per formare un task, così, ogni task è un punto nello skill space: $J^j \in$S. $J_i^j = 1$ denota che l'i-simo skill è richiesto dal j-simo task, $J_i^j = 0$ altrimenti. Così, si ottiene che $J^j = (J_1^j, J_2^j, \ldots, J_m^j)$.

User. E' considerato un set di persone (o esperti) P = $\{p_j$; j = 1, 2, . . . , n$\}$. Ogni esperto ha un sottoinsieme di skill nel suo profilo, così anche egli è rappresentato da un punto nello skill space: $p^j \in$ S. $p_i^j = 1$ denota che la j-sima persona ha l'i-simo skill, mentre $p_i^j = 0$ altrimenti. Così, si ottiene che $p^j = (p_1^j, p_2^j, \ldots, p_m^j)$.

Team. Ogni task necessita di essere assegnato ad un team di esperti. Sia $Q_j \subseteq$P il team assegnato al j-simo task. $q_i^j = 1$ denota che l'i-esimo skill è coperto dal j-simo team, cosicché $q^j = (q_1^j, q_2^j, \ldots, q_m^j)$. Per ogni team Q^j, il suo *team profile* $q_j \in$ S nell'*additive skill model* definisce l'esperienza (binaria) del team come la somma degli skill di ogni suo individuo:

$$q_i^j = \min \left\{ \sum_{p^l \epsilon Q^j} p_i^l , 1 \right\}_{i=1,\ldots,m}.$$

In altre parole, uno skill è coperto dal team se in esso esiste almeno un membro che ha quello skill. Riguardo il team per il task J^j è usata la notazione q^j e Q^j in modo intercambiabile.

Dato un team Q con profilo q assegnato al task J, è definito *cov*(q,J) ∈ {0, 1} dove è 1 se e solo se $q_i^j \geq J_i^j$, $\forall_i = 1, \ldots, m$ ed implica che il team Q copre il task J. Ciò accade se ogni skill richiesto dal task è posseduto dal team.

$\mathcal{P}$	Set di persone	*n*	Numero di persone
$\mathcal{S}$	Set di skill	*m*	Numero di skill
$\mathcal{J}$	Set di task	*k*	Numero di task
p^j	j-esima persona		
J^j	j-esimo task		
$d(p^j, p^{j\prime})$	Distanza tra la persona *j* e la persona *j'*		
$d(j, j')$	$d(p^j, p^{j\prime})$		
p_i^j	i-esimo skill della j-sima persona		
J_i^j	i-esimo skill del j-simo task		
Q^j	Team assegnato al j-simo task		
q^j	Skill profile del team j		
q_i^j	i-esimo skill profile del team j		
L(p)	Load della persona p		
L(Q)	Load del team Q		
c(Q)	Costo di coordinamento del team Q		
c(T)	Costo di coordinamento del sottografo T		
a(Q)	Costo di allocazione del team Q		
$a(p^j)$	Costo di allocazione della persona j		

Tabella 6 - Notazioni

Lo scopo, come accennato, è quello di formare un team capace di realizzare uno specifico task e allo stesso tempo ottimizzare due obiettivi, presumibilmente in conflitto: raggiungere una onesta allocazione dei carichi di lavoro complessivi tra le risorse e formare team che abbiano il più basso costo di coordinazione possibile. Ora queste nozioni verranno dettagliate in modo più preciso.

***Carico*.** La prima quantità da ottimizzare è il carico L(p) di un esperto *p*, che è definito come il numero dei task ai quali *p* partecipa: L(p) = |{j; $p \in Q_j$}|. In particolare, si è interessati alla minimizzazione del massimo carico tra tutti gli esperti.
***Costi di coordinamento*.** In pratica, sono i costi di coordinamento del lavoro di gruppo. Come già teorizzava il nobel Ronald Coase (1937), alla base dell'efficienza c'è spesso il coordinamento e l'incentivazione che non possono essere raggiunti gratuitamente, ma comportano dei costi "di transazione". Tali costi implicano il raggiungimento dell'efficienza allocativa e se, non giustamente dimensionati, limitano fortemente le ambizioni dell'impresa. Nel setting di riferimento, il coordinamento finalizzato alla collaborazione può vanificare i potenziali vantaggi ottenibili da gran parte delle competenze dei dipendenti.
I costi di coordinazione su un set di persone sono modellati tramite un social network, G=(P,E), ed è considerata una funzione di distanza metrica $d : E \rightarrow R^+$ sugli archi della rete. Questa funzione può modellare parametri quali: riferimenti alle interazioni passate, prossimità geografica, compatibilità nella collaborazione, distanza gerarchica nell'organigramma aziendale, e altro ancora. La definizione di funzione di distanza $d(\cdot,\cdot)$ è estesa ad ogni coppia di persone nella rete, considerando la somma delle distanze lungo il più breve cammino tra le due persone considerate.
Ci sono molti modi quindi per definire il costo di coordinamento di un team a partire dalla funzione *d* tra le coppie di persone nel social network. Prima di descrivere le specifiche misure dei costi di comunicazione, sono considerate due opzioni naturali relativamente alla tipologia del network su cui poggia l'infrastruttura.
1 - Implicity Connected Team (ICT). In questo modello non è richiesto che ogni team Q sia utile a costituire un grafo connesso; è richiesto solo che ci siano i path di comunicazione tra le persone nel team e che attraversino gli altri membri del social network (non necessariamente presenti in Q). In questo modello, per definire le persone come "compatibili", è realisticamente sufficiente l'esistenza di un breve concatenamento di "conoscenti".
2 - Explicity Connected Team (ECT). Questo modello segue l'approccio di *Lappas* (par. 3.4) e richiede che ogni team Q formi un grafo connesso usando il set di archi T={$(p_i, p_j) : p_i, p_j \in Q$}.
La differenza tra i due modelli sostanzialmente è data dal sottografo sul quale è calcolato il costo di coordinazione. Nel caso di ICT, tale costo, è calcolato su tutto il social network; nel caso di ECT, sul sottografo indotto dal team.
In entrambi i casi, per i costi di coordinazione, sono utilizzate le seguenti misure:

- *Steiner tree* - Steiner (Q) è il costo del *min cost Steiner tree* T che ha come nodi terminali i membri del team.
- *Diameter* - Diam (Q) = $max_{p^i, p^{i'} \in Q} d^j(i, i')$
- *Costo di coordinazione* - *c*(Q) che diventa *c*(T) se l'insieme degli archi T connette i membri del team dato.

3.6.2 Risoluzione multi-criterio

Assemblare team, tenendo conto dei costi di coordinamento e dei carichi di lavoro, conduce ad un problema con una funzione obiettivo bi-criterio, che in generale è difficile da analizzare. L'approccio principale che è adottato usualmente in letteratura, sui problemi di ottimizzazione bi-criterio, sbilancia lo sforzo d'analisi su una sola funzione obiettivo. Ossia si limita il costo di una funzione e si ottimizza l'altra. Questo consente di capovolgere il problema di ottimizzazione bi-criterio in un problema di ottimizzazione con una singola funzione obiettivo. Nel caso di nostro interesse, il costo massimo di coordinamento è fissato a B per ogni job J^je verrà invece ottimizzato il massimo carico di una persona, con l'idea ferma che B sia un vincolo sul costo di coordinamento. Il generico problema di ottimizzazione illustrato qui di seguito sarà chiamato BALANCED SOCIAL TASK

ASSIGNMENT e consisterà nel minimizzare il massimo carico di una persona mentre ogni job viene assegnato al team che coprirà tutti gli skill del job, con un costo di coordinazione vincolato da:

$$\begin{aligned} &\min \max_{i \in \mathcal{P}} L(p^i) \\ &\text{cov}(J^j, q^j) = 1 \qquad \forall\, j \in \mathcal{J} \\ &c(Q^j) \leq B \qquad \forall\, j \in \mathcal{J}. \end{aligned}$$

Un'altra possibilità che potrebbe essere presa in considerazione è data dal fissaggio del massimo carico minimizzando quindi il costo di comunicazione. Questa forzatura circa il setting online potrebbe risultare però innaturale. Infatti, visto che non si conosce a priori il numero di task, è difficile stabilire un limite sul massimo carico totale. Invece è più comprensibile proporre un livello accettabile di coordinazione e cercare di minimizzare il carico che impedisce ai nuovi task di giungere. E' auspicabile che, pesando in maniera differente il carico di lavoro e i costi di coordinamento, si possa trovare un giusto compromesso tra questi due parametri.

3.6.3 Gli algoritmi di bilanciamento del carico

Sono analizzati, di seguito, gli algoritmi che danno maggiori garanzie per questo problema, utilizzando differenti funzioni relative ai costi di coordinazione nel setting online. In particolare è studiato un algoritmo generico online per il BALANCED SOCIAL TASK ASSIGNMENT che può essere applicato ad una serie di funzioni sui costi di coordinamento.

(Q^*, T^*)	Soluzione ottimale per il problema SOCIAL TASK ASSIGNMENT
$(Q^{opt}, T^{opt}) = (Q_\lambda^{opt}, T_\lambda^{opt})$	Soluzione ottimale per il problema modificato con il parametro λ
$f^{opt}(\lambda) = \lambda a(Q^{opt}) + c(T^{opt})$	Costo ottimale per il problema modificato con il parametro λ
$(Q^{apx}, T^{apx}) = (Q_\lambda^{apx}, T_\lambda^{apx})$	Soluzione γ-approssimata per il problema modificato con parametro λ
$f^{apx}(\lambda) = \lambda a(Q^{apx}) + c(T^{apx})$	Costo γ-approssimato per il problema modificato con parametro λ
$\lambda^* = c(T^*) / a(Q^*)$	

Tabella 7 : Notazioni per l'analisi

L'algoritmo online opera come segue: subito dopo l'arrivo di un nuovo task J, viene costituito un team per J atto alla risoluzione di una istanza del problema adeguatamente definita (offline) con il SOCIAL TASK ASSIGNMENT.
Il SOCIAL TASK ASSIGNMENT, sottoproblema per il task J, consiste nel selezionare il team Q che minimizza una specifica funzione obiettivo sul costo di allocazione *a*(Q), vincolata dal costo di coordinamento *c*(Q). Il problema per il task J è definito come segue:

$$\begin{aligned} &\min_{Q} a(Q) \\ &\text{cov(J,q)} = 1 \\ &c(Q) \leq B. \end{aligned}$$

Il SOCIAL TASK ASSIGNMENT problem dipende dallo specifico costo di coordinazione *c*(Q) e dal costo di allocazione *a*(Q) che adottiamo. Come anticipato prima, saranno considerate due misure per Q: *Steiner* (Q) e *Diam*(Q).

Funzione del costo di allocazione. La funzione di allocazione *a*(.) usata nell'istanza del SOCIAL TASK ASSIGNMENT dipende dal carico individuale sulle persone, come risulta dalla precedente loro allocazione sui primi j-1 task. Differenti scelte sulla funzione del costo di allocazione influirebbero sulla performance di tutti gli algoritmi online. Esiste una serie di funzioni relative al costo di allocazione che è possibile considerare quali: la minimizzazione della taglia del team, il carico massimo, o il carico totale. E' mostrato che tali funzioni, in termini di bilanciamento dei carichi di lavoro, non sono tanto performanti quanto la funzione di costo l'*ExpLoad* che sarà descritta a breve.

La funzione *ExpLoad* nasce dalla ricerca dello scheduling online, ed è dimostrata essere efficiente quando il costo di coordinamento non viene preso in considerazione. La funzione *ExpLoad* è efficace ugualmente sul costo di allocazione anche quando viene data priorità al costo di coordinamento.

Sia L(p^i) il carico su una persona i quando il task J è giunto dalla rete, e sia Λ un valore appropriatamente scelto che dipende dal costo della soluzione ottima. La funzione *ExpLoad* è definita come segue:

$$a(\mathrm{Q}) = \sum_{p^i \in Q} (2n)^{\frac{L(p^i)}{\Lambda}}$$

Usando *ExpLoad* è garantito che l'algoritmo sia efficiente in pratica.

Ora sarà descritto un approccio che garantisce una soluzione per il BALANCED SOCIAL TASK ASSIGNMENT problem con un ottimo rapporto competitivo, nel caso di Steiner cost con team implicitamente connessi. Bisogna quindi fornire una soluzione al seguente SOCIAL TASK ASSIGNMENT problem:

$$\min_{Q} \sum_{p^i \in Q} a(p^i)$$
$$\mathrm{cov}(J,q) = 1$$
$$\mathrm{c}(Q) \leq B,$$

Dove definiamo: $a(p^i) = (2n)^{\frac{L(p^i)}{\Lambda}}$, tale che $a(\mathrm{Q}) = \sum_{p^i \in Q} a(p^i)$.

Il vincolo $c(\mathrm{Q}) \leq$ B specifica che lo Steiner-tree cost dei nodi nel team selezionato Q deve essere almeno B. Questo vincolo, insieme con quello di copertura *cov*(J,q) = 1, definisce un problema di group Steiner Tree. La connessione al group Steiner tree è stata considerata anche negli algoritmi di *Lappas*.

Nel problema group Steiner tree, sono date un network G=(V,E), dove |V|=n, con una funzione di distanza $d : \mathrm{E} \rightarrow \mathrm{R}^+$, e una famiglia G= $\{g_1, \ldots, g_l\}$ di un gruppo di nodi, con g_i<V. L'obiettivo è trovare un sottoalbero T di costo minimo di G che contenga almeno un nodo di ogni gruppo g_i. Questo problema ha un'approssimazione $\gamma = \mathrm{O}(ln^2 n \ln \ln n \ln l)$.

Il problema in oggetto è più complesso visto che abbiamo bisogno di trovare un algoritmo A che restituisca una coppia (Q^A, T^A) che non solo soddisfi i vincoli del group Steiner tree, ma che minimizzi anche la funzione del costo di allocazione *a*(Q) sui nodi restituiti. Sia (Q*, T*) la soluzione ottima per il nostro Social Task Assignment problem.

Algoritmo 5. Steiner coordination-cost

```
1.  Function findGroup(J)
2.  /* trova il miglior gruppo per il task J */
3.    λ_A ← binarySearch (J,0,upeprBound)
4.    /*Stima il limite ; trova il più grande valore al disotto del limite con
      un'accuratezza di ε. */
5.    (Q^A, T^A) ← findGroupForGivenLambda (J, λ_A)
6.    return (Q^A, T^A)

1.  Function binarySearch(J, λ_1, λ_2)
2.  /*Ritorna il più grande valore tra λ_1 e λ_2 per il quale c(Q) è sotto il
      limite, con un gap di accuratezza pari a ϵ . */
3.    if (λ_2 − λ_1 < ε · λ_2 )
4.     /* I due andpoiny sono molto vici; valore trovato*/
5.      return λ_1
6.    end if
7.    λ ← (λ_1 + λ_2) /2
8.    (Q_λ^apx, T_λ^apx) ← findGroupForGivenLambda (J, λ)
9.    if ((Q_λ^apx) < 2γB)
10.     return binarySearch(J, λ, λ_2)
11.   Else
12.     return binarySearch(J, λ_1, λ_2)
13.   end if

1.  Function findGroupGivenLambda(J, λ)
2.  /* Trova ilmigliore gruppo sul grafo modificato per il valore dato di
      λ */
3.    G' ← createModifiedGraph(λ)
4.    (Q_λ, T_λ) ← solveGroupSteinerTree (G', J)
5.    return (Q_λ, T_λ)

1.  Function createModifiedGraph(λ)
2.  /*Ritorna un istanza del grafo modificato usando il valore di λ */
3.    V' ← A copia di V
4.    E' = {e_i = (i, i'); i ∈ V, i'∈ V', ω(e_i) = c(i) · λ}
5.    i'∈ V' appartiene al gruppo j se l'utente i possiede lo skill
      j
6.    G' = (V ∪V, E ∪ E',w)
7.    return G'
```

La function *findGroup(J)* risolve il problema Social Task Assignment per il task J. La strategia per risolvere il problema Social Task Assignment ad alto livello, è la seguente:
(i) Prima è definita una famiglia di istanze del problema, chiamate istanze del problema modificato, parametrizzate da un valore $\lambda \in R^+$. Ognuna delle istanze del problema modificato sarà un istanza del *Group Steiner Tree problem*. E' possibile anche dimostrare che c'è un valore $\lambda=\lambda^*$ tale che la soluzione dell'istanza del problema modificato diventi una soluzione approssimata al problema originale del Social Task Assignment.

(ii) Sebbene sia difficile calcolare precisamente il valore λ*, di seguito è mostrato come determinare efficientemente un valore di approssimazione che produce una soluzione approssimata del problema originale.

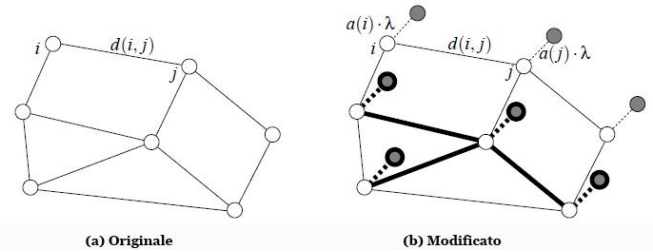

Figura 15: (a) Rete originale sulla quale sarà definito il Social Task Assignment problem. (b) Istanza del problema modificato sul quale è definito il group Steiner tree problem. In grassetto possiamo osservare un esempio di soluzione in (Q,T). Notiamo che il costo della soluzione è il costo di coordinamento c(T) del network originale, inoltre il costo dei nuovi archi è uguale al costo di allocazione a(Q).

(i) **Costruzione dell'istanza del problema modificato.**
Per un valore dato del parametro λ viene costruita la seguente istanza del problema modificato (vedi Figura15). Se il grafo iniziale è G = (*P*,*E*), viene creato un nuovo grafo G'= (*V*∪*V'*, *E*∪*E'*) con un nuovo nodo fittizio i'∈V' per ogni nodo i∈V, e con un nuovo arco e_i = (i,i') ∈ E' di costo λ*a*(i), ∀ i∈V.
Il problema Group Steiner Tree è definito sull'istanza del problema modificato, nel quale ogni skill richiesto dal task considerato trova corrispondenze nel gruppo e dove un nodo *i'* appartiene all'r-simo gruppo se il corrispondente esperto *i* possiede l'r-simo skill. Una soluzione al group Steiner tree sull'istanza del problema modificato è una copia (Q,T) dove Q è l'insieme dei nodi selezionati che coprono tutti i gruppi (skill) e T un sottoalbero che connette i nodi in Q. Il costo della soluzione (Q,T) sull'istanza modificata è data da

$$f(Q,T,\lambda) = \lambda_a(Q) + c(T).$$

Fissato un algoritmo randomizzato A, il costo della soluzione (Q^A, T^A) è definito come

$$f^A(\lambda) = \lambda_a(Q^A) + c(T^A),$$

dove $f^A(\lambda)$ è il costo atteso della soluzione di A.
Sia *opt*=opt_λ l'algoritmo originale per l'istanza del problema modificato tale che $f^{opt}(\lambda)$ è il costo della sua soluzione ottima (Q^{opt}, T^{opt}). Similmente, per un algoritmo γ-approssimato apx = apxλ, $f^{apx}(\lambda)$ sarà il costo della soluzione (Q^{apx}, T^{apx}) prodotto da apx. (Q^{opt} λ, T^{opt} λ) indica che il valore di λ per la soluzione è ottimo, se il valore è invece approssimato si scriverà (Q^{apx} λ, T^{apx} λ).
Ricordando che (Q*, T*) è la soluzione ottima al problema del Social Task Assignment, si può definire:

$$\lambda^* = c(T^*) \,/\, a(Q^*)$$

Ne consegue che il valore λ* è il valore di λ per il quale si desidera risolvere il problema considerato. La soluzione del problema modificato per λ = λ* fornisce una soluzione approssimata all'STA problem, secondo la seguente proposizione, di cui è omessa la dimostrazione.

PROPOSIZIONE 3. Le seguenti disequazioni sono vere:
$C(T\lambda*^{opt}) \leq 2B$;

$$A(Q\,\lambda *^{opt}) \leq 2a(Q^*).$$

(ii) Calcolare λ*. Fino ad ora è stato mostrato che conosciuto il valore λ*, si può definire l'istanza del problema modificato appropriata per risolverlo, questo può condurre ad una soluzione approssimata del problema originale (Social Task Assignment). Bisogna provare però che si è in grado di trovare il valore in modo efficiente. Attualmente, visto che non è possibile risolvere il Group Steiner Tree sull'istanza del problema modificato, non si può ancora calcolare l'esatto valore di λ*. E' possibile però stimare tale valore tramite una soluzione approssimata bi-criteria al problema originale. Per stimare il valore λ* è sfruttata l'informazione che la funzione $f^{opt}(\lambda)$ è una funzione monotona di λ (come mostrato nel prossimo lemma 4). Questo permetterà di stimare λ* usando la ricerca binaria. Inoltre risulta evidente che solo avendo una stima di λ* è possibile ottenere un'approssimazione bi-criteria.

LEMMA 4. Le seguenti affermazioni sono vere:

1. La funzione $f^{opt}(\lambda)$ non è decrescente in λ.
2. Si assume che $\lambda_1 < \lambda_2$. Da cui $a(Q^{opt}\lambda_1) \geq a(Q^{opt}\lambda_2)$ e $c(T^{opt}\lambda_1) \leq c(T^{opt}\,\lambda_2)$.

Essenzialmente questo Lemma consente di stimare λ* con la precisione richiesta. La descrizione esatta del suo utilizzo è nell'Algoritmo 5.
Lo studio si focalizza ora sul fatto che non è possibile risolvere ogni istanza del problema ottimamente quando si esegue la ricerca binaria, ma solo in maniera approssimata. Bisogna innanzitutto ricordare che ($Q^{opt}\lambda$, $T^{opt}\lambda$) è la soluzione ottima dell'istanza del problema modificato per il parametro λ (cioè, il valore che minimizza *f*(Q,T,λ)). Inoltre, per meglio comprendere l'algoritmo, è più facile descrivere una versione in cui si risolve ottimamente l'istanza del problema modificato per ogni valore di λ. Poi viene stimato il valore di λ* eseguendo una ricerca binaria. In ogni step della ricerca binaria è selezionato il valore di λ che è la media dei due punti finali λ_1 e λ_2, quindi viene risolta l'istanza del problema modificato. Ad ogni step è tenuto in conto che c($T^{opt}\,\lambda_1$) < B e c($T^{opt}\lambda_2$) ≥ B. L'algoritmo si ferma quando λ_1 e λ_2 sono vicini abbastanza, e la seconda affermazione del Lemma 4 garantisce che $\lambda_1 \leq \lambda^* \leq \lambda_2$. Visto che è pensabile risolvere il problema solo con un'approssimazione di γ, non si può calcolare il valore esatto di λ*. Ne consegue che la ricerca per trovare un valore che soddisfi tale relazione viene effettuata inserendo al suo posto un rapporto di approssimazione. A denota l'algoritmo find-Group tale che la soluzione ritorni (Q^A, T^A) e sia λ_A la stima di λ* calcolato da A. Le considerazioni su riportate sono alla base del seguente teorema, la cui dimostrazione, per brevità, è omessa. Il risultato stabilisce che l'algoritmo considerato è un algoritmo di approssimazione bi-criterio per il problema del Social Task Assignment con complessità (O(γ),O(γ)).
Teorema. Per ogni task J, l'Algoritmo findGroup(J) ritorna una soluzione (Q^A, T^A) tale che:

$C(T^A) \leq 2\gamma B$.
$A(Q^A) \leq (2\gamma/1\text{-}\varepsilon)\ a(Q^*)$. // con ε numero positivo infinitamente piccolo

Combinando una serie di Teoremi si deduce che esiste un algoritmo di approssimazione bi-criterio per il BALANCED SOCIAL TASK ASSIGNMENT problem (ricordando che *n* è il numero di persone, *m* il numero di skill, e *k* il numero di task):

> "*Esiste un O(ln² n ln ln nln m ln k) algoritmo competitivo per il Balanced Social Task Assignment problem, che, per ogni task, ottiene un costo di coordinamento Steiner-tree limitato da O(B ln² n ln ln n ln m).*"

Tempo di esecuzione. E' facile verificare che la complessità dell'algoritmo 5 sia polinomiale. Questo perché: (i) ogni invocazione di *findGroupForGivenLambda* è eseguita in tempo polinomiale; (ii) la funzione binarySearch è invocata un numero polinomiale di volte.
Per vedere che (ii) è corretta, basta notare che il parametro *upperBound* alla linea 3 dell'algoritmo è settato sul più grande valore possibile del rapporto c(T)/a(Q). Inoltre $c(T) \leq \sum_{p^j,p^l \in P,} d(p^j, p^l)$ e il limite inferiore ad a(Q) è pari ad 1 (il peso su ogni persona è 0 quando è allocato il primo task e la taglia del team è 1). L'affermazione consegue poi anche da quante volte è invocata la *binarySearch* che al più potrà essere logaritmicamente limitata inferiormente per una costante ε. Osserviamo comunque che, sebbene polinomiale, il costo dell'algoritmo può essere in pratica molto alto, a causa del costo non lineare dell'algoritmo group Steiner tree. Per questo motivo, vengono prese in considerazione le euristiche che implementano findGroupForGivenLambda meno esigenti dal punto di vista computazionale.

3.6.4 Gli algoritmi sul costo di coordinamento

Per il Diameter Coordination Cost (*Diam*) è seguito un differente e più semplice approccio, presentato nell'algoritmo 6. Similmente alla precedente funzione sul costo di coordinamento, si cerca di trovare un team di utenti Q con diametro almeno γB (è fissato γ = 2) che minimizza a(Q). In questo caso, anziché costruire una nuova istanza del problema, per combinare carico di lavoro e costo sociale della funzione, il problema viene risolto direttamente.

Algoritmo 6. Diameter algorithm

```
1.  Function findGroup(J)
2.  /* Trova il migliore gruppo per il task J*/
3.    r ← arg min_{i∈S} Σ_{j∈P} p_i^j
4.    /* r adesso è uno degli skill più rari*/
5.    Q^A ← 0
6.    foreach (ℓ tale che p_r^ℓ =1)
7.      U ← { p^j; d(p^ℓ, p^j) ≤ B}
8.      Q ← greedyWeightedSetCover (U)
9.      if (a(Q) < a(Q^A))
10.       Q^A ← Q
11.     end if
12.   end foreach
13. return Q^A
```

I risultati sono riassunti nel seguente teorema.

> TEOREMA. Esiste un algoritmo per il problema Balanced Social Task Assignment che per ogni task ottiene un diameter coordination cost limitato da 2B in tempo O(ln m ln k).

Tempo di esecuzione. Il tempo di esecuzione dell'algoritmo è ovviamente polinomiale: il principale ciclo (linee 6-12 dell'Algoritmo 6) è eseguito al più *n* volte, mentre ogni esecuzione richiede un ciclo dell'euristica greedy (polinomiale) per il set cover pesato.
Per il Sum-of-Distances Coordination Cost (*Sum*) è possibile applicare una tecnica simile, e definire un'appropriata istanza del problema modificato. Per risolverlo è definita una appropriata classe di istanze del problema modificato, formulato come un problema di programmazione convessa che viene risolto usando il potente metodo del rounding randomizzato. Questo consente di ottenere risultati allineati agli ultimi due teoremi enunciati.

Finora sono stati sviluppati degli algoritmi che probabilmente, approssimati alla soluzione ottima nei casi online, sono eseguiti ugualmente in tempo polinomiale. Tali algoritmi sono stati adattati per il modello con la connessione implicita dei team (ICT).
Per il modello con connessione esplicita dei team ECT, il problema è di difficile risoluzione: la difficoltà di risoluzione dello Steiner Coordination Cost, è tanto hard quanto il problema del Group Steiner Tree con nodi e archi pesati.
In particolare, si vuole formulare un problema di ottimizzazione che indirizzi lo studio agli obiettivi multi criterio, cercare di minimizzare il carico massimo e, allo stesso tempo, si vuole mantenere il costo di coordinazione limitato da una costante B.
L'algoritmo teorico usato per il Group Steiner Tree problem, anche se fornisce garanzie asintotiche, non è molto pratico. Perciò negli esperimenti sono state applicate alcune semplici euristiche. Un'euristica potrebbe essere l'approccio usato da *Lappas*, che è basato sulla creazione di un'altra istanza di grafo, sulla quale viene risolto il problema dello Steiner Tree. In sintesi, per ogni skill (o gruppo), è creato un nuovo nodo connesso a tutti gli esperti che possiedono quello skill aggiungendo un costo *D* grande abbastanza. Poi viene calcolato lo Steiner tree che connette gli skill richiesti. Per l'ultimo task utilizziamo l'euristica del minimum Spanning Tree.

3.6.5 Il Set Cover

E' stato sviluppato anche un ulteriore algoritmo, chiamato euristica set-cover di cui vengono presentati i dettagli nell'Algoritmo 7.

Algoritmo 7. Set-cover Steiner algorithm

```
1.  Function findGroupForGivenLambda-setCover(J, λ)
2.  /* Trova il migliore gruppo per il task J, usando l'euristica set-cover*/
3.    Q ← 0
4.    T ← 0
5.   while (∃i s.t. (J_i − q_i)^+ > 0)
6.     mostCostEffective ← 0
7.     foreach (p^j ∈ 𝒫 \ Q )
8.        gain(p^j) ← Σ_{i=1}^{m} 1_{{p_i^j=1 ∧ J_i=1 ∧ q_i= 0}}
9.        loss(p^j) ← costo in G per ogni aggiunta p^j in Q
10.       costEffectiveness ← gain(p^j)/loss(p^j)
11.       if (costEffectiveness > maxcostEffectiveness)
12.          maxcostEffectiveness ← costEffectiveness
13.        mostcostEffective ← j
14.       end if
15.    end foreach
16.  T ← T ∪ "shortest path da p^mostcostEffective a T"
17.  Q ← Q ∪ {p^mostcostEffective} /* Questo vale per il modello ICT.
     Per il modello ECT invece aggiungiamo tutti gli espert al path */
18.  end while
19.  return (Q,T)
```

I costi di allocazione (appropriatamente parametrizzati da λ) saranno descritti dai nodi del social network mentre i costi sociali dagli archi. Visto che si vuole produrre soluzioni vicine all'ottimo, è assunto che la soluzione dell' algoritmo condivide almeno un esperto con il team della soluzione ottima. Ciò è ottenuto effettuando una ricerca tra i nodi-radice relativi a tutti gli esperti che posseggono gli skill meno frequenti del task (cioè lo skill posseduto dal più piccolo numero di

esperti). Siano Q e T il team e lo Steiner Tree costruito fino a questo punto dell'esecuzione dell'algoritmo. Alla linea 8, $gain(p^j)$ è settato sul numero degli skill richiesti posseduti dall'esperto p^j che non è ancora coperto da Q. (La notazione 1_A denota la funzione indicatore per A). Per vedere se ci si trova in questo caso, è constatato che l'esperto p^j contribuisce all'i-simo skill se e solo se possiede lo skill ($p_i^j = 1$). Lo skill quindi è richiesto ($J_i = 1$), ma non è ancora coperto ($q_i = 0$). Alla linea 9 $loss(p^j)$ è settato al costo che occorre per aggiungere l'esperto p^j al team Q come descritto in seguito. Poi, ad ogni step dell'algoritmo, l'esperto più *cost-effective* è aggiunto a Q, dove cost-effectiveness è definito dal rapporto $gain(p^j)/loss(p^j)$. L'algoritmo termina quando tutti gli skill del task sono coperti.

Gli algoritmi riguardanti le diverse nozioni di costo di coordinazione, per i modelli ICT e ECT, differiscono solo nella definizione della funzione di costo $loss(p^j)$.

$l^s(p^j, p^{j\prime})$ denota la lunghezza del path di costo minimo $p^s(p^j, p^{j\prime})$, nel social network N che connette p^j a $p^{j\prime}$, con i costi di allocazione (appropriatamente scalati) di tutte le persone sul percorso, escludendo $p^{j\prime}$. Definiamo quindi $loss(p^j) = \min_{p^{j\prime} \in Q} l^s(p^j, p^{j\prime})$.

Nel modello ICT, una volta trovato la persona p^j con il miglior cost-effective, il path $p^s(p^j, p^{j\prime})$ è aggiunto a T e l'esperto p^j è incluso al team Q.

Nel modello ECT, un volta trovato l'esperto p^j con il miglior cost-effective, il path $p^s(p^j, p^{j\prime})$ è aggiunto a T e il team Q è esteso con l'inclusione di tutti gli esperti sul path, affinché il team risultante sia connesso al sottografo. In questo caso, in $l^s(p^j, p^{j\prime})$ deve essere seriamente considerato anche il costo dell'inclusione di questi esperti. Nel modello ECT, gli esperti sul path $p^s(p^j, p^{j\prime})$, non solo possono contribuire alla connessione con gli altri, ma possono anche fornire skill addizionali per il task. Tali path saranno considerati coperti da questi esperti che faranno quindi parte del team generato.

Un altro approccio ai problemi di ottimizzazione bi-criterio è calcolare l'insieme delle soluzioni ottimali di Pareto, cioè quelle soluzioni che non hanno migliori risultati se eseguite mediante altre decisioni, sia sui costi di allocazione che di coordinamento. L'insieme delle soluzioni ottimali di Pareto solitamente formano un curva convessa che definisce un chiaro trade-off tra costi di allocazione e costi sociali. E' auspicabile che, un punto nello spazio, sulla curva di Pareto, non abbia coordinate uguali rispettivamente al minimo costo di coordinamento e il minimo costo di allocazione, per ogni soluzione.

I due approcci sono argomenti molto trattati in letteratura e gli algoritmi per questo problema possono essere usati per risolvere altri problemi. Se l'ottimizzazione è relativa ai problemi NP-hard, si può solo sperare di trovare, in tempo polinomiale, soluzioni quasi ottimali di Pareto. Esse, probabilmente, non saranno più performanti (se consideriamo tutti e due gli obiettivi) rispetto ad altre soluzioni con un limite (Papadimitriou, 2000).

L'idea degli algoritmi prima analizzati è che il giusto trade-off tra allocazione e coordinamento è dato dal rapporto λ^* tra costi di coordinamento e di allocazione nella soluzione ottima. Il problema viene perciò ridotto alla risoluzione del Social Task Assignment problem con costo di allocazione scalato da λ^*. Un algoritmo A restituisce una soluzione (Q^A, T^A) che minimizza $f^A(\lambda) = \lambda a(Q^A) + c(T^A)$. Visto che, in pratica, non viene considerato il limite sul costo di coordinazione, non è calcolato il profilo sulle soluzioni di Pareto. Si è proceduto invece con l'ottimizzazione di $f^A(\lambda)$ per i differenti fattori scalari di λ. Questo ha consentito di identificare il miglior trade-off tra costi di allocazione e costi sociali. Tuttavia, per risolvere (approssimativamente) il problema di ottimizzazione, è sufficiente osservare solo i valori di λ che sono richiesti dagli algoritmi 5 e 6.

3.7 Il modello basato sulla densità

Come visto fin qui, esistono molti metodi atti alla risoluzione del nostro problema che considerano però assunzioni troppo restrittive al punto che, spesso, non possono essere implementati in scenari realistici. L'obiettivo del modello che ci accingiamo qui a descrivere è quello di considerare il problema del team formation in un setting realistico e presentare una nuova formulazione basata su sottografi densamente popolati. Tale formulazione consente la modellazione di molti aspetti naturali come l'inclusione di team leader designati e/o di gruppi di esperti, la diminuzione della taglia o più generalmente del costo del team, l'introduzione di vincoli sulla *località* del team (in senso geografico o sociale).
Questa formulazione conduce alla versione generalizzata del problema classico del sottografo denso con cardinalità vincolata (DSP), che è un problema NP-hard e che ha molte applicazioni nell'analisi del social network. In questa sezione, presentiamo un nuovo metodo per risolvere (approssimativamente) il DSP generalizzato (GDSP). Il metodo, chiamato **FORTE**, è basato sulla soluzione di una relazione equivalente continua del GDSP. La soluzione che nasce dal succitato metodo, ha garanzie di qualità e soddisfa sempre i vincoli del GDSP. Gli esperimenti mostrano che la formulazione proposta (GDSP) è applicabile nella modellazione di un ampio range di problemi sul team formation e che esso produce team di alta qualità ossia coerenti e compatti. Con l'ausilio della relazione LP di GDSP, è mostrato che il metodo fornisce soluzioni vicine all'ottimo.
In questo paragrafo si evidenzia soprattutto che molti metodi sviluppati in passato non considerano esplicitamente le connessioni tra gli esperti da un punto di vista sociale e professionale. Ciò induce anche ad una discussione sull'aspetto sociologico del team formation e la sua influenza sull'evoluzione delle comunità, aspetto che tralasciamo in questo studio per ovvi motivi di brevità.

3.7.1 Formulazione

I criteri che misurano l'efficienza dei team, che si trovano in letteratura oggi, sono basati sui cammini minimi, sulla densità, e sul costo del minimum spanning tree del sottografo indotto dal team. Tenendo ben presente il concetto di densità presentato nel Cap. 3.1.2, è possibile affermare che i team ben connessi hanno valori di densità alti.
I metodi basati sulla minimizzazione del diametro (i maggiori cammini minimi tra ogni coppia di nodi) o sul costo dello spanning tree hanno il principale vantaggio che i team prodotti sono sempre connessi (purché il social network sottostante sia connesso). Sia il diametro che lo spanning tree formati per tale scopo non sono flessibili ai cambiamenti (aggiunta o cancellazione di archi) nel social network.
Dall'altro lato, massimizzando la densità si potrebbe ottenere un team il cui grafo è disconnesso. Questo accade soprattutto quando ci sono piccoli gruppi di persone fortemente connesse tra loro, ma connesse in modo sparso con il resto del grafo.
Esistono metodi che non fanno riferimento a nessuna delle due forti assunzioni di cui sopra, oppure non sono in grado di includere vincoli più intuitivi, quale ad esempio, il limite sulla taglia totale del team. L'obiettivo di questa sezione è quello di considerare il problema del team formation in uno scenario più realistico e presentare una formulazione innovativa basata sulla generalizzazione del problema del sottografo più denso, modellando molti aspetti realistici, quali: l'inclusione di un gruppo di esperti, la possibilità di limitare la taglia o il costo del team, l'introduzione del concetto di località del team.
Considerato il problema del Team Formation, già ben definito e formalizzato in precedenza, è introdotto il concetto di peso $w_{ij} \in$W simmetrico e non negativo, che connette due esperti *i* e *j*, e che nella fattispecie riflette il livello di compatibilità tra essi. L'insieme di skill è dato da A= $\{a_1, \ldots, a_p\}$. E' assunto che ogni esperto abbia uno o più skill. La matrice non negativa M $\in R^{n \times p}$ descrive il

livello di skill di tutti gli esperti, per ogni tipo di skill. Il livello di skill è definito su una scala di valori continui. Se un esperto *i* non ha skill *j*, allora M_{ij}=0. Inoltre è usata la notazione $M_j \in R^{n\times 1}$ per la *j*-sima colonna di M, cioè il vettore dei livelli di skill corrisponde allo skill *j*. Un task T è dato dall'insieme delle triple $\{(a_j, k_j, l_j)\}_{j=1}^{p}$, dove $a_j \in$ A, specificando che, per finire il task, è richiesto almeno k_j e al più l_j skill di a_j.

Problema generalizzato del team formation. Dato un task T, il problema generalizzato è definito come la ricerca di un team C⊆V di esperti che massimizzano la compatibilità collaborativa e soddisfano i seguenti vincoli:

- **Inclusione di un gruppo specifico**: un predeterminato gruppo di esperti S⊂V potrebbe essere incluso in C.
- **Skill richiesti**: per finire il task T, sono richiesti almeno k_j e al più l_j skill di a_j.
- **Limite sulla taglia del team**: la taglia del team potrebbe essere più piccola o uguale a b, cioè |C| ≤ b.
- **Vincoli di budget**: il budget totale per finire il task è limitato da B, cioè $\sum_{i\in C} C_i \leq$ B, dove $C_i \in R_i$ è il costo sostenuto dall'esperto *i*.
- **Vincolo sulla Distanza**: la *distance* (misurata secondo la funzione simmetrica non negativa *dist*) tra ogni coppia di esperti in C potrebbe non essere più grande di d_0, cioè, *dist*(u,v) ≤ d_0, $\forall$ u,v ∈ C.

3.7.2 I vincoli

In contrasto ai metodi preesistenti, è consentito che il limite superiore su ogni skill sia uguale alla taglia totale del team. Considerando la matrice di skill solo come binaria (allo stesso modo del paragrafo precedente), tutto si tradurrebbe in un limite superiore e inferiore sul numero di esperti richiesti per ogni skill. Usando i vertici pesati, invece sarà possibile codificare più vincoli generici, come, ad esempio, il limite uguale al budget totale del team. Non è banale estendere i metodi esistenti per includere ogni limite superiore. Al netto delle nostre conoscenze, questo sembra il primo metodo che integra nel problema del team formation i vincoli sulla taglia del team. E' auspicabile inoltre che ci sia un secondo vincolo affinché il team formation sia più realistico. Questo tipo di setting consente agevolmente di selezionare un gruppo di esperti intorno al quale poi formare il team. Un'altra importante generalizzazione è l'inclusione del vincolo di distanza per ogni generica funzione di distanza. Un vincolo può essere così utilizzato per rinforzare il concetto di località del team, per esempio in senso geografico (la distanza potrebbe essere il tempo di attraversamento) oppure la distanza nella rete in senso sociale. Le mutue incompatibilità tra i membri del team, per esempio al livello personale, potrebbero essere rappresentate da una grande distanza tra gli esperti che non sono compatibili, in modo tale da non essere messi insieme nello stesso team.

Misure di compatibilità collaborativa. La misura di compatibilità collaborativa è una forma generalizzata di densità sui sottografi, definita come:

$$densità(C) := \frac{assoc(C)}{vol_g(C)} = \frac{\sum_{i,j\in C} w_{ij}}{\sum_{i\in C} g_i}$$

Dove w_{ij} è il peso non negativo dell'arco tra i e j e vol_g(C) è definita come $\sum_{i\in C} g_i$, con g_i peso positivo del vertice *i*. Settando g_i = 1 per ogni $i \in$ V, sarà restituita la formulazione originale di densità. La relazione assoc(C) = vol_d(C) – cut(C,V\C) denota il grado del vertice *i* dove cut(A, B) := $\sum_{i\in A, j\in B} w_{ij}$ e d_i=$\sum_{j=1}^{n} w_{ij}$.

3.7.3 La densità

L'obiettivo basato sulla densità gode di proprietà quali la completa monotonicità e la robustezza. Nel caso dell'obiettivo basato sulla densità, se un arco porta ad un incremento (a causa di una nuova collaborazione) o ad una sottrazione (a causa di nuove incompatibilità) della densità del sottografo, allora a quest'arco sarà necessariamente richiesto un incremento o un decremento del proprio valore. Nel caso dell'obiettivo basato sul diametro questo non vale ed inoltre sarà maggiore l'impatto di piccoli cambiamenti nel struttura del grafo.
La densità generalizzata che è usata in questo contesto porta inoltre ad una modellazione maggiormente libera che consente di dare pesi agli esperti secondo le loro esperienze. Dando piccoli pesi a chi ha grossa esperienza, è possibile ottenere una soluzione che non solo soddisfi pienamente gli skill richiesti ma prediliga una composizione di team con membri maggiormente competenti (cioè quelli che hanno pesi più piccoli).
Usando la notazione prima introdotta, un'istanza del problema del team formation basata sulla densità generalizzata può essere formulata come:

$$\begin{aligned} & \max_{C \subseteq V} \frac{\boldsymbol{assoc}(\boldsymbol{C})}{\boldsymbol{vol_g}(\boldsymbol{C})} \\ \text{vincolato da:} \quad & S \subseteq C \\ & k_j \leq vol_{M_j}(C) \leq i_j \quad \forall\, j \in \{1, \dots, p\} \\ & |C| \leq b \\ & vol_c(C) \leq B \\ & \text{dist}(u,v) \leq d_0 \quad \forall\, u,v \in C \end{aligned} \qquad (3.5)$$

I vincoli di upper bound sulla taglia del team e il budget possono essere riscritti come vincoli sugli skill e possono essere incorporati perciò nella matrice degli skill M. Così senza perdita di generalità, da ora in poi, per questioni di brevità, i vincoli di budget e quelli di taglia saranno omessi. Inoltre essendo richiesto che S sia parte della soluzione, è possibile assumere che $dist(u,v) \leq d_0$, Per ogni $u,v \in S$, diversamente il problema succitato è irrisolvibile. Il vincolo di distanza implica anche che $u \in V$ per qualche $dist(u,s) > d_0$, per qualche $s \in S$, non può essere parte della soluzione; altrimenti accade che tali vertici possono essere eliminati senza mutare la soluzione del problema (3.5).
La formulazione (3.5) è la versione generalizzata del problema classico del sottografo più denso (DSP) che ha molte applicazione nelle analisi dei grafi. La più semplice versione di DSP consiste nel trovare il sottografo più denso (senza qualche vincolo sulla soluzione), che può essere risolto ottimamente in tempo polinomiale. Il problema del più denso k-sottografo, il quale richiede che la soluzione contenga esattamente *k* vertici, è notoriamente un problema hard ed è stato dimostrato che non ammette soluzioni in tempo polinomiale. Recentemente è stato dimostrato che il problema del sottografo più denso con un upper bound sulla taglia è hard esattamente come il problema del più denso k-sottografo. Comunque il problema del sottografo più denso con vincolo di lower bound ha un'algoritmo 2-approssimato. E' basato sulla soluzione di una sequenza del problema del sottografo più denso senza vincoli. E' stato anche dimostrato che esiste una relazione di programmazione lineare per questo problema che raggiunge la stessa garanzia di approssimazione.
Recentemente è stata considerata la versione generalizzata del problema del sottografo più denso con vincoli di lower bound nel contesto del problema del team formation:

$$\max_{C \subseteq V} \frac{assoc(C)}{vol_g(C)}$$

vincolato da: $S \subseteq C$

$$vol_{M_j}(C) \geq k_j \qquad \forall\, j \in \{1, \dots, p\} \qquad (3.6)$$

Dove M è la matrice binaria di skill. E' stato esteso il metodo greedy di Khuller (1996) e viene mostrato che è realizzato con una garanzia 3-approssimata per alcuni casi specifici del problema. Recentemente è stata migliorata la garanzia di approssimazione dell'algoritmo greedy per il problema (3.6) su fattore 2. La complessità temporale dell'algoritmo greedy è O(kn^3), dove n è il numero di esperti e k := $\sum_{j=1}^{m} k_j$ è il numero minimo di esperti richiesto.

Integrazione diretta di vincoli su sottoinsiemi. Il vincolo di sottoinsieme può essere integrato in un obiettivo lavorando direttamente sul sottografo G' indotto dall'insieme di vertici V' = V\S. Ogni C⊂ V che contiene S può essere scritto come C = A∪S, per A⊂V'. Adesso è possibile riformulare il problema del team formation sul sottografo G', non prima di aver introdotto la notazione *m*=|V'| e di aver assunto che le prime *m* entrate di V sono le prime anche in V'.

I termini del problema (3.5) possono essere riscritti come

assoc(C) = assoc(A) + assoc (S) + 2cut(A,S),

= $vol_d(A)$ – cut(A,V\A) + assoc(S) + 2cut(A,S)

= $vol_d(A)$ – cut(A,V'\A) + assoc(S) + cut(A,S)

$vol_g(C) = vol_g(A) + vol_g(S)$.

Per di più, è possibile scrivere: cut (A,S) = $vol_{d^S}(A)$, dove $d_i^S = \sum_{j \in S} w_{ij}$ denota il grado del vertice *i* ristretto al sottoinsieme S nel grafo originale. Usando l'abbreviazione, μ_s = assoc(S), $v_s = vol_g(S)$, $assoc_s(A) = vol_d(A)$ – cut (A,V'\A) + μ_s + $vol_{d^S}(A)$, riscriviamo il problema del team formation (3.5) come

$$\max_{A \subseteq V', A \neq 0} \frac{assoc_s(A)}{vol_g(A) + v_s} \qquad \text{(GDSP)}$$

vincolato da: $k_j \leq vol_{M_j}(A) \leq l_j \qquad \forall\, j \in \{1, \dots, p\}$

$dist\,(u, v) \leq d_0 \qquad \forall\, u,v \in A$

in cui per tutti gli $j = 1, \dots, p$, i limiti sono aggiornati tali che $k_j = k_j - vol_{M_j}(S)$, $l_j = l_j - vol_{M_j}(S)$ e dove è stata usata l'assunzione: dist(u, s) ≤ d_0, per ogni μ∈V e per ogni s∈S.

Il vincolo A≠0 è stato introdotto per ragioni tecniche dovute dalla formulazione del problema continuo. L'equivalenza tra il problema (GDSP) e la (3.5) segue dalla considerazione che sia S che l'insieme A*∪S (dove A* è una soluzione ottima per GDSP), dipendono da chiunque abbia maggiore densità.

Non esiste un algoritmo greedy con una garanzia di approssimazione per risolvere il problema (GDSP). Invece di progettare un algoritmo con approssimazione greedy per questo problema di ottimizzazione discreta, è stato derivato un problema *equivalente* di ottimizzazione continua, come dimostrato più avanti. Il problema discreto quindi verrà riformulato nello spazio continuo mentre sarà preservata l'ottimalità delle soluzioni nel campo discreto. Il motivo di questo approccio è dovuto al fatto che la formulazione continua è più flessibile e permette di scegliere in un più grande insieme di metodi per la sua soluzione, rispetto al campo discreto. Sebbene il problema continuo risulti hard alla pari del problema originale discreto, i recenti sviluppi nell'ambito dell'ottimizzazione continua consentono di trovare molto efficientemente una soluzione ottima.

3.7.4 Derivazione di FORTE

Formation Of Realistic TEams (FORTE), usando la relazione di continuità, è proposto come (DSP) generalizzato. FORTE è stato dedotto in tre step:

i. Viene derivato un problema equivalente discreto senza vincoli dal problema del team formation (GDSP) tramite un approccio exact penalty (Di Pillo, 1986).
ii. E' derivata una relazione equivalente continua del problema senza vincoli, usando il concetto di estensioni di *Lovasz* (Lee, 2013) .
iii. Viene calcolata la soluzione del problema continuo usando il metodo *RatioDCA* dalla programmazione frazionaria (Setzer, 2011).

Il problema equivalente senza vincoli. Una tecnica generale nell'ottimizzazione vincolata è quella di trasformare il problema vincolato in uno equivalente senza vincoli, con l'aggiunta, all'obiettivo, di un termine di penalità, controllato da un parametro $\gamma \geq 0$. Il termine di penalità è zero se i vincoli sono soddisfatti ad un input dato, altrimenti è strettamente positivo. La scelta del parametro di regolarizzazione γ, avendo un obiettivo di basso valore, influenza il tradeoff tra i vincoli soddisfatti. Grossi valori di γ tendono a rinforzare il soddisfacimento dei vincoli. E' mostrato che per il problema del team formation (GDSP) esiste un valore di γ che garantisce il soddisfacimento di tutti i vincoli.
Il termine di penalità per i vincoli del problema GDSP è definito come segue:

$$\text{pen(A)} := \begin{cases} \sum_{j=1}^{p} \max\{0, vol_{M_j}(A) - l_j\} & \\ +\sum_{j=1}^{p} \max\{0, k_j, & \\ -\ vol_{M_j}(A)\} & \\ +\sum_{u,v\in A} \max\{0, \text{dist}(u,v) - d_0\} & A\neq 0 \\ 0 & A=0 \end{cases}$$

La funzione di penalità succitata vale zero solo quando A soddisfa i vincoli; altrimenti è strettamente positiva e si incrementa all'aumentare della non realizzabilità. Lo speciale trattamento dell'insieme vuoto è un tecnicismo trattato in seguito dall'estensione di Lovasz. Per la stessa ragione, anche i termini costanti μ_s e v_s sono sostituiti rispettivamente con $\mu_s$unit(*A*) e $v_s$unit(*A*), dove unit(*A*):=1, A≠0 e unit(0)=0.
Esiste un problema equivalente senza vincoli per il problema di ottimizzazione vincolato (GDSP), infatti è dimostrabile che: il problema vincolato (GDSP) è equivalente al problema senza vincolo

$$\min_{0\neq A\subseteq V} \frac{vol_g(A)+v_s\text{unit}(A)+\gamma\text{pen(A)}}{assoc_s(A)} \qquad (3.7)$$

Per $\gamma > \frac{vol_d(V)}{\theta}$ $\frac{vol_g(A_0)+v_s}{assoc_s(A_0)}$, dove A_0 è qualche insieme realizzabile del problema (GDSP) tale che $\text{assoc}_S(A_0)>0$ e θ è il minimo valore di non fattibilità, cioè *pen(A)*$\geq \theta$, se A è non realizzabile.

Il problema equivalente continuo. Ora è derivata una relazione completamente continua del problema (3.7). Questo ci condurrà alla minimizzazione del problema su R^m, che possiamo manipolare più facilmente rispetto al problema originale discreto. La connessione tra lo spazio discreto e quello continuo è realizzata attraverso i limiti. Dato un vettore f $\in R^m$ si può definire l'insieme

$$A_i := \{j \in V \mid f_j \geq f_i\} \qquad (3.8)$$

dalla limitazione di *f* al valore f_i.

Con le estensioni di Lovasz è possibile transitare dalle funzioni sugli insiemi alle funzioni sullo spazio continuo.
Estensione di Lovasz - Sia R : 2^v -> R un insieme di funzioni con R(0)=0, e sia $f \in R^m$ ordinato in senso ascendente $f_1 \leq f_2 \leq \ldots \leq f_m$. L'estensione di Lovasz $R^L : R^m \rightarrow$ R è definita da

$$R^L(f) = \sum_{i=1}^{m-1} \mathrm{R}(A_{i+1})(f_{i+1} - f_i) + \mathrm{R}(\mathrm{V}f_1).$$

R^L (1_A) = R(A) per tutti A⊂V , cioè R^L è indotta un estensione di R da 2^v verso R^V (|V|=m). In seguito, dato un insieme di funzione R, la sua estensione di Lovasz sarà denotata da R^L.
Mostrando l'equivalenza con la programmazione di un insieme frazionale, possono essere trovati risultati più generali. In particolare è dimostrabile che:

Il problema discreto senza vincoli (3.7) è equivalente al problema continuo.

$$\min_{f \in R_+^{V'}} \frac{vol_g^L(f) + v_s \mathrm{unit}^L(\mathrm{A}) + \gamma pen^L(f)}{assoc_S^L(f)} \qquad (3.9)$$

f_j^*
Per ogni γ≥0. Inoltre, l'ottimo è limitato da un minimizzatore f* ∈ R_+^m,

$$A^* := \min_{A_i := \{j \in V' | f_j^* \geq f_i^*\}} \frac{vol_g(A_i) + v_s + \gamma \mathrm{pen}(A_i)}{assoc_s(A_i)},$$

produce un insieme A* che è ottimo per il problema (3.7).

Inoltre a corollario del precedente, è possibile affermare che: il problema del team formation (GDSP) è equivalente al problema (3.9) se γ è scelta secondo la condizione data da (3.7).
Mentre il problema continuo è hard come il problema discreto originale, una recente idea sull'ottimizzazione del problema continuo permette di derivare un algoritmo molto efficiente per ottenere una soluzione localmente ottima.

3.7.5 L'algoritmo e il metodo FISTA

Qui di seguito sarà descritto un algoritmo per risolvere approssimativamente il problema continuo di ottimizzazione (3.9). L'idea è quella di utilizzare il concetto che il problema di ottimizzazione frazionato (3.9) ha una speciale struttura: può essere scritto come un particolare rapporto tra la differenze di funzioni convesse, cioè nella forma:

$$\min_{f \in R_+^V} \frac{R_1(f) - R_2(f)}{S_1(f) - S_2(f)} := Q(f) \qquad (3.10)$$

Dove le funzioni R_1, R_2, S_1 e S_2 sono funzioni omogenee positivamente convesse con il numeratore e denominatore non negativi. Questa riformulazione poi consente di usare un recente metodo di ordinamento chiamato RatioDCA. Allo scopo di trovare la forma esplicita delle funzioni convesse, per primo c'è bisogno di riscrivere il termine di penalità come pen(A) = pen_1(A) – pen_2(A), dove

$$\mathrm{pen}_1(\mathrm{A}) = \sum_{j=1}^{p} vol_{M_j}(A) + \sum_{j=1}^{p} k_j \ \mathrm{unit}(\mathrm{A}),$$
$$pen_2(\mathrm{A}) = \sum_{j=1}^{p} \min\{l_j, vol_{M_j}(A)\} + \sum_{j=1}^{p} \min\{k_j, vol_{M_j}(A)\} - \sum_{u,v \in \mathrm{A}} \max\{0, \mathrm{dist}(u,v) - d_0\}.$$

Usando questa decomposizione di pen(A), è possibile ora ridurre le funzioni R_1, R_2, S_1 e S_2 come

$$R_1(f) = vol_p^L(f) + \sigma \max_i\{f_i\}$$
$$R_2(f) = \gamma\, pen_2^L(f)$$
$$S_1(f) = vol_d^L(f) + vol_{ds}^L(f) + \mu_s \max_i\{f_i\}$$
$$S_2(f) = cut^L(f)$$

dove $\rho := g + \gamma \sum_{j=1}^{p} M_j$, $\sigma := v_s + \gamma \sum_{j=1}^{p} k_j$, $pen_2^L(f)$
pen_2 (A) denota l'estensione di Lovasz e

$$vol_h^L(f) = \langle (h_i)_{i=1}^m, f \rangle \text{ dove h} \in R^n,$$
$$cut^L(f) = \frac{1}{2} \sum_{i,j=1}^{m} w_{ij} |f_i - f_j| .$$

E' dimostrabile che: usando le funzioni R_1, R_2, S_1 e S_2 definite prima, il problema (3.9) può essere riscritto nella forma (3.10). Le funzioni R_1, R_2, S_1 sono convesse e positivamente omogenee, e R_1-R_2 e S_1-S_2sono non negative.
La riformulazione del problema nella forma (3.10) consente di applicare una modifica del RatioDCA, metodo per la minimizzazione locale di un obiettivo della forma (3.10) su R^n.

Algoritmo 8. Minimizzazione locale (di un rapporto sulla differenza delle funzioni convesse.

1. Inizializzazione: $f^0 \in R_+^m, \lambda^0 = Q(f^0)$
2. Repeat
3. $f^{l+1} = \arg\min_{u \in R_+^m, ||u_2|| \leq 1} R_1(u) + \lambda^l S_2(u) - \langle u, r_2(f^l) + \lambda^l s_1(f^l) \rangle$
4. $\lambda^{l+1} = Q(f^{l+1})$
5. until $\frac{\lambda^{l+1} - \lambda^l}{\lambda^l} < \epsilon$

Data un'inizializzazione f_0, l'algoritmo di cui sopra risolve una sequenza di problemi di ottimizzazione convessa (linea 3). Non c'è bisogno di una descrizione esplicita dei termini $s_1(f)$ e $R_2(f)$, ma solo degli elementi del loro sottodifferenziale, ossia rispettivamente $S_1(f) \in \partial S_1(f)$ e $r_2(f) \in \partial R_2$ (f). Il problema convesso (alla linea 3) avrà la forma

$$\min_{f \in R_+^m} \frac{\lambda^l}{2} \sum_{i,j=1}^{m} w_{ij} |f_i - f_j| + \langle f, c \rangle + \sigma \max_i\{f_i\} \quad (3.11)$$

dove c = ρ-$r_2(f^l)$ – $\lambda^l s_1(f^l)$. (3.11) è un problema non uniforme ma esiste un problema duale uniforme, che è possibile osservare qui di seguito. Il problema (3.11) è equivalente a:

$$\min_{||\alpha|| \leq 1} \min_{f \in S_m} ||P_{R_+^m}(-c - \frac{\lambda^l}{2} A_\alpha - \sigma v)||_2^2 ,$$

dove A: $R^E \to R^V$ con $(A_\alpha)_i := \sum_j w_{ij} (\alpha_{ij} - \alpha_{ji})$, $P_{R_+^m}$ denota la proiezione sull'ordinata positiva e S_m è il simplesso $S_m = \{v \in R^m \mid v_i \geq 0, \sum_{i=1}^m v_i = 1\}$.
Il problema duale uniforme può essere risolto molto efficientemente usando il più scalabile metodo chiamato FISTA, che ha un rapporto di convergenza di $O(\frac{i}{k^2})$, dove k è il numero degli step eseguiti in FISTA. La computazione principale in FISTA consta di una matrice di moltiplicazioni. Visto che tipicamente, il social network è sparso, questa operazione costa O(m), dove m è il numero degli elemento non nulli in W.

RatioDCA, produce una sequenza f^l strettamente decrementale, cioè Q(f^{l+1}) < Q(f^l), oppure termina. Questa è una tipica proprietà dei metodi localmente veloci scalati su grossi networks. Notiamo che la convergenza verso l'ottimo globale di (3.10) non può essere garantita dalla natura non convessa del problema.
Sussiste comunque la seguente garanzia di qualità per il problema del team formation.
E' possibile dimostrare che, sotto determinate condizioni, RatioDCA termina dopo un'iterazione oppure produce A_{f*} che soddisfa tutti i vincoli del problema GDSP di team formation e

$$\frac{assoc_s(A_{f*})}{vol_g(A_{f*})+ v_s} > \frac{assoc_s(A_0)}{vol_g(A_0)+ v_s},$$

dove A_0 è un insieme fattibile per il problema (GDSP), *f** è il risultato del RatioDCA dopo l'inizializzazione del vettore 1_{A_0}, A_{f*} è l'insieme che risulta dalla segmentazione ottima di *f**.
Relazione LP di GDSP. Il problema del team formation basato sulla densità, dopo l'integrazione del vincolo di sottoinsieme, è riscritto come il seguente GDSP:

$$\max_{A \subseteq V'} \frac{assoc_s(A)}{vol_g(A) + v_s} \qquad (3.12)$$

vincolato da: $k_j \leq vol_{M_j}(A) \leq l_j \quad \forall\, j \in \{1, \dots, p\}$
$dist(u,v) \leq d_0 \quad \forall\, u,v \in A$

In 3.12 non è richiesto alcun vincolo addizionale come A≠0, che invece è stato aggiunto a GDSP. In questa sezione è mostato che esiste una relazione di programmazione lineare (LP) per tale problema. La relazione LP può essere risolta ottimamente in tempo polinomiale e fornisce un limite superiore al valore ottimo di GDSP.
La seguente relazione LP è un'istanza del Problema Generalizzato sulla Densità del Sottografo (3.12)

$$\max_{t \in R,\ f \in R^{V'}, f \in R^{E'}} \sum_{i,j=1}^{m} w_{ij}a_{ij} + 2\,\langle d^S, f\rangle + t\mu_s \qquad (3.13)$$

vincolato da:

$tk_j \leq \langle M_j, f\rangle \leq tl_j$	$\forall\, j \in \{1, \dots, p\}$
$f_u + f_v \leq t,$	$\forall\, u,v : dist(u,v) > d_0$
$t \geq 0,\ \alpha_{ij} \leq f_i,\ \alpha_{ij} \leq f_j$	$\forall\, (i,j) \in E'$
$0 \leq f_i \leq t,\ \alpha_{ij} \geq 0$	$\forall\, (i,j) \in E'$
$\langle g, f\rangle + tv_s = 1$	

dove V' = V\S, E' è l'insieme degli archi indotti da V'.
La soluzione f^* di LP (3.13) è, in generale, non integrale, cioè, $f^* \notin \{0,1\}^{V'}$. Si possono usare tecniche standard di rounding randomizzato oppure di segmentazione ottimale per derivare una soluzione integrale da f^*. Comunque, il risultato di una soluzione integrale non restituisce necessariamente un sottoinsieme che soddisfa i vincoli di (3.12). Nel caso particolare ci sono solo vincoli di lower bound, cioè, il problema (3.6) con limite f_i^*, produce un sottoinsieme A_i non vuoto (nel caso peggiore l'insieme pieno V'), soddisfacendo tutti i vincoli di lower bound.
La relazione LP (3.13) non è valida se ci riferiamo ai vincoli di limite inferiore. Infatti, $f \in R^m$ non soddisfa i vincoli di upper bound di LP (3.13), cosa invece realizzabile solo dopo la ripesatura di *f*

stessa senza dover cambiare l'obiettivo di LP. Questo implica che è sempre possibile trasformare la soluzione del problema senza vincoli in una soluzione fattibile, ammesso che ci siano solo vincoli di upper bound. Comunque, in presenza di lower bound o vincoli di sottoinsieme (come il rescaling), la (3.6) non produce una soluzione fattibile, perciò la relazione LP è utile sull'istanza (3.12) con almeno un lower bound o un sottoinsieme di vincoli (cioè, v_s>0).

3.8 Il modello basato sulle capacità

Gli studi precedentemente illustrati hanno indicato che la cooperazione diviene effettiva in presenza di connessioni sociali. Perciò, la concreta selezione del team richiede che i suoi membri debbano essere socialmente collegati così come si auspica che ci sia una equa suddivisione dei task tra i membri, tale che nessun utente sia sovraccaricato dall'assegnamento.
In questo paragrafo è studiato in che modo questi team possano essere formati su un social network, tenendo conto però delle capacità dei singoli attori. Inoltre, tenendo presente che i problemi di formazione di team sono NP-hard, verranno introdotti 2 algoritmi di approssimazione efficienti per trovare team vicini all'ottimo. Saranno considerate poi alcune estensioni del problema e descritti degli algoritmi adattati alla risoluzione in questo scenario. Verranno quindi illustrati i risultati degli esperimenti su dataset relativi a vari social network, con comparazione rispetto *GitHub*, che realizza una riduzione dei costi di collaborazione di circa il 40%! Infine, per dimostrare la generalità delle tecniche, è presentato un set addizionale di esperimenti sul dataset DBLP. I risultati sperimentali riveleranno che gli algoritmi, in generale, raggiungono almeno il 20% di risparmio sul costo totale, se paragonati ad altre euristiche.
Lo studio a cui fa riferimento questa sezione, anche se molto vicino a quello effettuato da Theodoros Lappas (2009), differisce dai numerosi modelli presentati in precedenza soprattutto per il fatto che in essi il problema fondamentale è l'allocazione di risorse ed inoltre solo alcuni considerano il social network definito tra gli utenti. Gli autori di tale modello (Datta, 2012) hanno pensato di selezionare i team di utenti da un social network, inoltre tra i membri (che saranno 'socialmente efficienti' in termini di collaborazione) sussisterà una suddivisione equa di task tale che nessun utente venga sovraccaricato dall'assegnamento e soprattutto a nessun utente siano assegnati task al di là delle proprie capacità. Sarà pertanto formulato il classico problema del team formation, pensandolo su di un social network e saranno quindi aggiunti dei vincoli di capacità sugli utenti. Questi sembrano essere aspetti della problematica non ancora sufficientemente studiati in letteratura.

3.8.1 Notazioni

Saranno descritte brevemente le notazioni utili a comprendere il modello di seguito illustrato.
Rappresentando, al solito, il social network come un grafo non direzionato, gli archi vengono pesati dal costo della funzione $w : E \rightarrow R^+$, indicante il costo della collaborazione tra gli utenti vicini nella rete. Perciò, per gli utenti *u* e *v* tali che (u,v) ∈ E, quanto più *w*(u,v) è grande, tanto è meno probabile che essi possano collaborare bene. La funzione di costo *w* può essere stimata dall'analisi delle tracce delle precedenti collaborazioni tra gli utenti.
Ogni utente *v*∈V è skillato con un set di item, dove gli item, sono le abilità utili ad un dato progetto. Ad esempio, nel caso di un progetto software: i linguaggi di programmazione, i sistemi operativi, le notazioni sistemistiche, insomma le conoscenze necessarie. I_v denota il set di item per gli utenti *v* mentre || indica l'universo di tutti gli item nella rete, cioè $|| = U_{v \in V} I_v$. c_v invece denota le capacità dell'utente *v*, cioè il massimo numero di item che possono essere assegnati a *v* senza rischio di sovraccarico.
In fine, un task T è definito come un sotto insieme di item$\{i_1, i_2, \ldots, i_k\} \subseteq ||$. Ad un utente *u* può essere assegnato un item *i* dal task solo se egli è skillato per quel task ossia: $i \in I_u$.

Verranno adesso descritti i vincoli di fattibilità ed il "costo di collaborazione sociale".
Un team U ⊆ V è detto essere ***fattibile*** per un dato task T se soddisfa i seguenti vincoli:

- *Vincoli di covering*: C'è un assegnamento valido *v* dell'item in T verso gli user in U tale che ogni item i ∈ T è assegnato a qualche user ψ(i) tale che i ∈ $I_{\psi(i)}$.
- *Vincoli di packing*: Per ogni user u ∈ U, il numero totale di item assegnato ad u non eccede la sua capacità, cioè $\sum_{i \in T : \psi(i)=u} 1 \leq c_u$.

In virtù della funzione di costo *w*(•), è possibile calcolare il costo di collaborazione tra ogni coppia di utenti *u* e *v*. Se u e v sono *friends*, cioè (u,v) ∈ E, il costo sarà semplicemente w(u,v); altrimenti, sarà il costo del più corto percorso che connette u e v. Comunque, se un team ha più di due utenti, non sarà banale estendere questa definizione per il costo di collaborazione. In aiuto pervengono le seguenti misure del costo di collaborazione, prese in prestito dal lavoro di Lappas.

- *Diameter Cost*: Dato un social network G = (V,E,w), il diameter cost di un team U⊆V è definito come il costo massimo tra ogni coppia di user in U.
- *Steiner Cost*: Dato il social network G = (V,E,w), lo steiner cost di un team U⊆V è definito come il minimo costo di un albero che connette tutti gli utenti in U. Più formalmente, lo steiner cost per un team U è stabilito come il costo del minimo steiner tree che connette tutti gli user in U.
- *Bottleneck Cost*: Dato il social network G(V,E,w), e un team U⊆V, il bottleneck cost è definito come il minimo costo di uno spanning tree U, dove il costo di un tree è definito come il massimo peso di ogni arco nell'albero.

Questi modelli di costo sono intuitivi e colgono alcune importanti proprietà di un team. E' possibile osservare che, mentre il diametro non ammette coppie di utenti e il bottleneeck cost non ammette utenti vicini per i quali esiste un grande costo di collaborazione, lo steiner cost, d'altro canto, è una misura aggregata che cerca di ridurre soprattutto il costo di collaborazione, quindi considera in qualche modo le relazioni tra gli utenti.

3.8.2 Formulazione del problema

Chiarite le poche notazioni utili, ora è possibile formulare il problema.
In un problema di team formation, un utente che ha preso in carico un task, dovrà collaborare con gli altri utenti appartenenti al suo social network per costituire un team, cosicché, essi possano insieme raggiungere l'obiettivo. Formalmente, il problema, può essere definito nel modo seguente: *"Dato un social network G(V,E,w), un utente v che partecipa ad un task T, bisogna trovare un gruppo di utenti U⊆V, tale che il team U∪{v} sia fattibile per il task e minimizzi il costo di collaborazione sociale (misurato in diameter, steiner o bottleneck cost)."*
Ci si riferirà al problema del team formation con diameter collaboration cost, quale *MinDiamTeam*; Similarmente, le varianti con steiner e bottleneck collaboration cost sono riferite rispettivamente al *MinAggrTeam* e *MinMaxTeam*.
In generale è assunto che l'utente *v* specifichi anche una lunghezza minima di salto *h* cosicché gli user che sono lontani da esso più di *h* salti, non saranno considerati dal processo di selezione del team. Valori tipici di *h* sono 1 (amici), 2 (amici di amici) e ∞ (svincolato, cioè l'intera rete).
Ovviamente è garantito che l'utente *v* (che crea il task) sia un membro del team. Questo scenario è rappresentativo della collaborazione sociale reale soprattutto in un ambiente di sviluppo software, dove l'utente che crea il progetto contribuisce anche ad esso. Comunque, questa assunzione non è essenziale per lo specifico problema; l'algoritmo, infatti, può essere facilmente adattato nel caso in cui *v* non sia considerato parte della soluzione, ma sia solo un decision maker esterno ad essa!

3.8.3 Algoritmi

In questo paragrafo saranno illustrati gli algoritmi di approssimazione per calcolare i team vicini all'ottimo secondo il modello sulle capacità e sul costo quindi verrà descritto un algoritmo esatto per il MinMaxTeam.

LEMMA. MaxItems: dato un task T=$\{i_1, i_2, \ldots, i_k\}$ e un team di utenti U⊆V, il massimo numero di item che possono essere assegnanti ad U tali che a nessun utente in U vengano assegnati più item della loro capacità, può essere calcolato in tempo polinomiale.

Come corollario a tale Lemma, può essere dimostrato che U forma un team fattibile per il task T.

Algoritmo 9. Algoritmo MinDiamSol per la minimizzazione del diameter cost.

---

1. **Input**: Social Network G(V,E,*w*); User *v*; Task T=$\{i_1, \ldots, i_k\}$; lunghezza massima di salto *h*.
2. **Output**: Team *U*⊆V ; allocazione di T in U.
3. Pre-processa il grafo per escludere gli user che sono lontani più di *h* salti da *v*.
4. Trova il più piccolo raggio r' tale che MAX-ITEMS(T_v (r')) = *k*;
5. Sia $Team_v$ il team cercato;
6. return $Team_v$;

OPT denoterà la soluzione ottima e |OPT| il suo costo. L'Algortimo 9 descrive lo pseudo-codice per il problema del *MinDiamTeam*. L' idea è quella di individuare il valore di |OPT| e decidere se c'è un team fattibile il cui costo è una buona approssimazione del valore immaginato. È improbabile trovare la soluzione ottima tra tutti i possibili valori di |OPT|, visto che il problema in oggetto è NP-Hard. Per questo, invece di minimizzare il diameter cost del team, si cerca piuttosto di minimizzare la distanza tra il root user (cioè *v*) ed i membri del team.

Assunto che *r* sia il costo immaginato, si usa $T_v(r)$ per denotare l'insieme degli utenti la cui distanza da *v* è al più *r* cioè T_v(r) = $\{u|d(v,U)\leq r\}$, dove d(v,u) è il costo dello shortest path che connette *v* ad *u*, e v ∈ T_v (r), per tutti i valori di *r*.

L'obiettivo è quello di determinare il minimo valore di r cosicché il set di utenti T_v(r) formi un team fattibile per il task dato.

E' dimostrabile che *MinDiamSol* è un'algoritmo 2-approssimato per il problema MinDiamTeam.

Complessità temporale. Il team fattibile MinDiamSol può essere trovato effettuando una ricerca binaria sul diametro del grafo, che richiede O(log n_h) invocazioni del Lemma 2 dove n_h denota il numero di nodi che sono distanti *h* salti dallo user root *v*. Perciò, la complessità nel caso peggiore è O(kn_h^2(log n_h + log k)log n_h). L'analisi però suggerisce che il tempo di esecuzione di questo algoritmo è molto minore rispetto a questo caso peggiore.

L'algoritmo per il problema MinAggrTeam è leggermente più complicato rispetto al caso che contempla il diameter. L'algoritmo *MinAggrSol* inizia con la trasformazione del social network in un grafo semplice G' perchè trovare il team nel grafo G' è concettualmente più facile che farlo nel grafo originale. Ne consegue che l'algoritmo sceglie, con tecnica greedy (trovando l'ottimo locale), i nodi da G', basandosi su una misura di utilità, finché il team fattibile non viene trovato. Tali passaggi sono tradotti in pseudocodice nell'algoritmo *MinAggrsol* come di seguito illustrato

Algoritmo 10. Algoritmo MinAggrSol per la minimizzazione dello steiner cost.

1. **Input**: Social Network G(V,E,*w*); User *v*; Task T={i_1, … , i_k}; lunghezza massima di salto *h*.
2. **Output**: Team *U*⊆V ; allocazione di T in U.
3. Pre-processa il grafo per escludere gli user che sono lontani più di *h* salti da *v*.
4. G'(V,E',λ) ← AugmentGraph(G).
5. $cover_v := \{v\}$;
6. while $cover_v$ non fattibile do
7. ẍ ← argmax$_{x\{V\setminus cover_v\}}\{\frac{MAXITEMS(cover_v \cup \{x\}) - MAXITEMS(cover_v)}{\lambda(v,x)}\}$;
8. $cover_v \leftarrow cover_v$ {ẍ}
9. end while
10. $Team_v$ ← MinSteinerTree ($cover_v$,G); */* minimum cost tree in g che connette tutti gli user in $cover_v$ */*
11. return $Team_v$;

La procedura del *Augmentgraph* (linea 4 di MinAggrSol) crea un grafo G'(V, E', λ) nel modo seguente: per ogni vertice *u*∈V \{*v*}, aggiunge un arco (*v*,*u*) ∈ E'. Il costo dell'arco (*v*,*u*) ∈ E', cioè λ(*v*,*u*), è definito come il costo del più corto percorso tra *u* e *v* in G. Concettualmente la costruzione produce un primo livello di albero con gli utenti *v* (radice) e con tutti gli altri utenti al livello 1. Un team fattibile in G' corrisponde al sotto-albero di G' con utente *v* come radice. Il team ottimo in G' è denotato da OPT_λ.

Sebbene G' strutturalmente è più semplice rispetto a G, riducendo il problema a quello del set cover, può essere dimostrato che, nonostante sia rimossa la ricerca di OPT_λ il problema è ancora NP-hard. Per trovare un'approssimazione ad OPT_λ, si inizia con un team che contiene solo una parte degli utenti *v*. Ad ogni step, si aggiungono (in modalità greedy) gli user da G' al team solo se esso massimizza l'*utility*. L'utility di un nodo *u*∈V\{v} è definito come il rapporto tra il beneficio marginale dell'aggiunta di *u* al cover (in termini di numero di item addizionali che esso copre) e il suo costo $\lambda(u,v)$. La costruzione evidenziata nel Lemma su descritto (*MaxItems*) è usata per derivare i valori di utilità. Una volta trovato un team fattibile, viene costruito uno steiner tree di costo minimo che connette tutti gli utenti del team e ritorni come soluzione il team stesso. Sebbene il minimum steiner tree è un problema NP-hard, ci sono algoritmi il cui costo di computazione ha un fattore di approssimazione pari a 2. Il lemma seguente mostra che il costo della soluzione ritornato dal *MinAggrSol* non è lontano dal costo della soluzione ottima.

TEOREMA.

E' dimostrabile che l'Algoritmo *MinAggrSol* ottiene un fattore di approssimazione O(k logk) al problema MinAggrTeam, dove k è il numero di item richiesto dal task.

Inoltre sussistono i seguenti tre Lemmi:

- (L1) il tempo di esecuzione di OPT_λ è al più di O(k) volte il costo di OPT.
- (L2) la funzione *MaxItems*(S) su di un set di utenti S⊆V è sottomodulare.
- (L3) l'algoritmo greedy che seleziona gli insiemi basati sull'utility massima, raggiunge un fattore di approssimazione O(logk) rispetto al problema sottomodulare del set cover.

Vengono considerati: l'insieme di item E, il set dei requisiti T = {i_1, i_2,…, i_k}, e una collezione di sottoinsiemi di C come l'insieme degli utenti che posseggono un sottoinsieme di item da T. Visto che MaxItems è submodulare e i requisiti sono tutti soddisfatti, è possibile concludere che, attraverso gli

step 8-11 dell'Algoritmo 10, è possibile ottenere una soluzione O(logk)-approssimata ad OPT_λ. Combinando questo con il L1, si giunge al risultato del teorema sopra enunciato.

Complessità temporale. In *MinAggrSol*, la taglia di $cover_v$ è al più k. Perciò, calcolando il minimum cost team si va incontro ad un costo di $O(k^3 \log k + n_h k^2 + m)$ (step 4-9).

Lo steiner tree computation, alla linea 10, può essere eseguito in tempo addizionale di O(m+n log n). Praticamente, MinAggrSol scala almeno linearmente rispetto all'incremento della taglia della rete (cioè *m* e *n*) e al numero degli item (cioè *k*).

L'Algoritmo *MinMaxSol* descrive a grandi linee lo pseudo codice risolutivo per il problema MinMaxTeam. Si presuppone che il team ottimo non contenga nessun arco con costo maggiore di *t* e che vengano rimossi gli archi con costo alto. Successivamente, viene identificato il componente del grafo che contiene un team fattibile; ciò perché ogni team che abbraccia alcuni componenti multipli deve usare un arco di costo maggiore a *t*. In fine si cerca il minimo valore di *t* tale che questo componente (contenente la root *v*) risulti un team fattibile. Può essere dimostrato facilmente che la soluzione ritornata da MinMaxSol è ottima.

Il valore minimo di *t* (linea 4) può essere trovato tramite una ricerca binaria sull'insieme degli archi nel grafo, che richiede $O(\log m_h)$ invocazioni del Lemma succitato (*MaxItems*), dove m_h è il numero degli archi nel grafo dopo lo step di pre-prosessing (cioè lo step 3 nel *minMaxSol*). Per cui il tempo totale di complessità è $O(k m_h^2 (\log n_h + \log k) \log m_h)$.

Fino a questo punto si era assunto che lo user root fosse stato ben specificato e che esso fosse parte della soluzione. Questa restrizione può essere comunque aggirata nel seguente modo: basta ripetere gli algoritmi per la scelta del nodo radice e far restituire il team che è di costo minimo. Questa operazione preserva i fattori di approssimazione dell'algoritmo ma incrementa la complessità temporale di un fattore costante *n*.

Allo stesso modo del caso "diameter", segue il risultato per lo Steiner. Sia *v* un nodo che appartiene alla soluzione ottima OPT. Considerando l'iterazione nel caso in cui *v* è selezionato come radice, viene ripetuto *MinDiamSol* per tutti i vertici radice.

In OPT, la distanza di tutti gli altri nodi da *v* è al più |OPT|; perciò, se *r* è scelto uguale ad |OPT| poi l'insieme degli utenti $B_v(r)$ dovrà contenere un team fattibile. Perciò, $r \leq |OPT|$. Applicando il succitato Teorema e, considerando che si sta ritornando il costo minimo del team relativo alla selezione dei vertici della radice, ne consegue il risultato desiderato.

Algoritmo 11. Algoritmo MaxSol per la minimizzazione del team bottleneck cost.

1. **Input**: Social Network G(V,E,*w*); User *v*; Task T={$i_1, \dots, i_k$}; lunghezza massima di salto *h*.
2. **Output**: Team $U \subseteq V$; allocazione di T in U.
3. Pre-processarre il grafo per escludere gli user che sono più lontani di *h* hop da *v*.
4. for t = 0 → *maxweight (E)* do
5. Rimuovi tutti gli archi $e \in E$ tale che $w(e) > t$;
   Sia C il componente che contiene *v*;
6. If MAXITEMS(*C*) = *k* return team;
7. end for

3.8.4 Estensioni

In questo paragrafo vengono descritte alcune estensioni del problema di riferimento ed è spiegato brevemente come gli algoritmi sopra riportati possano essere modificati per trattare tali casi.

Costo relativo agli user. In una situazione reale, gli utenti sono spesso retribuiti in rapporto alla loro partecipazione ai task. In tale scenario, l'obiettivo del problema del team formation non è solo trovare i team che minimizzano il costo della collaborazione sociale ma anche tener conto del costo relativo alla retribuzione degli lavoratori. Un approccio potrebbe essere quello di definire il costo di un team di utenti U nel seguente modo: $\alpha \sum_{\mu \in U} \pi_\mu + \beta f(U)$, dove π_μ è il prezzo pagato all'user $u \in U$ e $f(\bullet)$ rappresenta il costo di collaborazione sociale (come visto nel par. 3.8.1), dove α, β sono parametri del problema. L'obiettivo è quello di trovare il team con minor costo totale.

Allo scopo di tener conto della funzione di costo modificata, il grafo del social networking è preprocessato nel modo seguente:

- Per ogni utente u, è creato un nuovo nodo u' e aggiunto un arco (u,u') con costo $\alpha\pi_\mu$ cioè $w(u,u') = \alpha\pi_\mu$.
- Nello step successivo, ogni costo sugli archi del grafo originale viene moltiplicato per un fattore β. Cioè il costo dell'arco (u,v) è settato a $\beta \bullet w(u,v)$.
- In fine, vengono effettuate le seguenti assegnazioni:
  - $I_{u'} \leftarrow I_u$,
  - $c_{u'} \leftarrow c_u$,
  - $I_u = 0$,
  - $c_u = 0$.

Con questo apposito pre-processo, è possibile usare gli algoritmi *MinDiamSol* e *MinAggrSol* sul grafo modificato per calcolare le soluzioni approssimate e lasciando immodificati i fattori di approssimazione.

Item multipli. E' possibile considerare anche il seguente scenario: il task è un multi-set di item della forma di $T = \{(i_1, n_1), (i_2, n_2), \dots, (i_k, n_k)\}$, dove $\{i_1, i_2, \dots, i_k\}$ è l'insieme degli item che hanno bisogno del task ed n_j è la molteplicità dell'item i se $i \in I_\mu$; inoltre la somma di tutti gli item assegnati ad u non deve superare il limite rappresentato dalla propria capacità.

- 1° Il Lemma verrà modificato al fine di poter considerare le molteplicità degli item.
- 2° Nella costruzione del flusso di rete, le capacità degli archi (l_i, r_u) saranno settati ad n_i, dove i è un item nel task e u è un utente. Inoltre, gli archi della forma (s, l_i) avranno capacità n_i.

Può essere verificato che se il massimo flusso ha valore $\sum_{i \in T} n_i$, allora esiste un team fattibile. Con questa modifica della procedura MaxItems, è possibile applicare gli algoritmi MinDiamSol e MinAggrSol, ottenendo team approssimati. E' possibile mostrare che, mentre l'algoritmo MinDiamSol realizza un fattore di approssimazione pari a 2, l'algoritmo MinAggrSol ha un rapporto di approssimazione di $O(\log k \sum_{i \in T} n_i)$.

3.8.5 Sperimentazioni

Introduciamo brevemente il data-set su cui sono stati effettuati gli esperimenti. Innanzitutto è stata ristretta la selezione dei data-set ai soli network on-line. Concentrando l'attenzione su alcuni social-network (*Twitter, Facebook,..*) ci si è resi conto che essi non erano adatti alla sperimentazione perché presentavano una gamma povera di collaborazioni, mentre ci si è accorti che i cosiddetti "coding sites" erano più adatti all'obiettivo. Tra essi, GitHub serve una larga parte di progettisti software ai quali fornisce le caratteristiche di un social networking (messaging, attività di streams, links friendship). Sebbene ci siano altri coding sites (come per esempio *SourceForge, Google Codes*…) si è scelto GitHub per vari motivi: differentemente da altri siti, GitHub fornisce social links attraverso i

quali gli user possono connettersi con tuti gli altri; le API di GitHub consentono di estrarre grosse quantità di dati velocemente; si sta diffondendo rapidamente tra i programmatori. Inoltre, GitHub offre un nuovo modello di collaborazione sociale nel quale ogni utente può "forkare" una copia del progetto software posseduto dai suoi friend. Il contributo dell'utente a questa copia forkata può essere poi unito all'originale ed insieme utilizzati dagli altri. A GitHub, che consiste di 135346 utenti e 905000 progetti software, è stata fatta una istantanea nel mese di Agosto 2011. Tale data-set corrisponde al 15% del volume di GitHub e si espande su di un periodo di 4 anni circa. Per ogni progetto, è stata effettuata una selezione relativa a linguaggi di programmazione, liste di contribuzioni, logs e report bug. Similmente, per ogni utente, è stata selezionata la lista dei suoi followers e dei rispettivi progetti creati e a cui hanno contribuito.
Per gli esperimenti, sono stati utilizzato i seguenti due dataset:

1. D1 che consiste di 135346 utenti e 905000 progetti software e molti meta-dati circa gli utenti e i progetti (cioè *commit logs*, team composition etc..). Questo data set è usato per stimare i parametri degli algoritmi considerati.
2. D2 è usato come data point test. Questo data set consiste di un nuovo set di 470 progetti collezionati durante il periodo Settembre-Dicembre 2011 (cioè protraendo sul filo temporale i dati di D1).

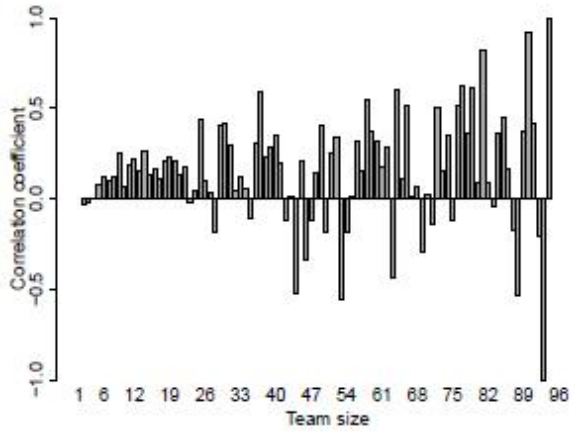

Figura 16 : Coefficienti di correlazione tra il numero di commesse di un progetto e il numero di archi sociali nel team per differenti taglie di team.

Sono stati analizzati i *commit logs* di 10000 progetti su di un periodo quindi di circa 4 mesi ed è stato effettuato lo studio di un interessante modello che analizza le relazioni tra tra l'attività del progetto (misurata in termini di numero di commesse) e la natura "sociale" del team corrispondente. Per questo esperimento, i team vengono raggruppati in base alla loro taglia (numero di collaboratori), inoltre è riportato un coefficiente di correlazione tra la "natura sociale" del team e il numero delle commissioni del progetto. Sono state usate due semplici misure per valutare la natura "sociale" del team: a) numero di archi sociali presenti nel team ; b) numero dei componenti connessi nel sottografo indotto dai membri del team.

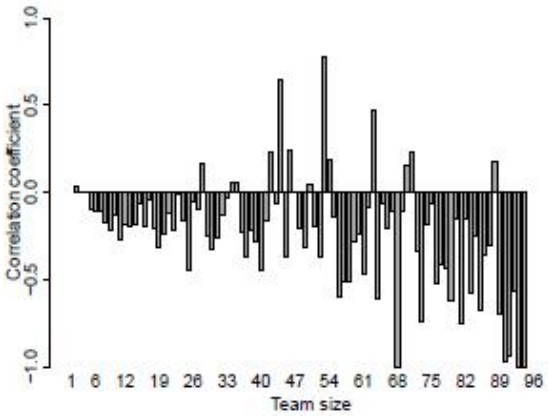

Figura 17 : Coefficienti di correlazione tra il numero di commesse di un progetto e il numero delle componenti connesse nel sottografo indotto dai membri del team per differenti taglie di team.

La Figura16 e la Figura17 mostrano il risultato in questo esperimento. E' possibile osservare che i team con più archi (perciò più sociali) sono più attivi in termini di numero di componenti connesse nei loro progetti. Inoltre il coefficiente di correlazione resta positivo attraverso quasi tutte le differenti taglie del team in Figura16. Similarmente, i team con più grande numero di componenti connesse (quindi meno sociali) sono meno attivi e risulta un coefficiente di correlazione negativo attraverso quasi tutte le differenti taglie dei team (Figura17). Questi risultati suggeriscono che in GitHub, la natura sociale di un team incide fortemente con l'attività del progetto.

3.8.5.1 La fase di Pre-processing

Social Network. In virtù della lista di follower, è costruito un social network con gli utenti inclusi nel cammino desiderato. Visto che la collaborazione, in natura, è simmetrica, gli archi sono trattati come non direzionati. Per ogni coppia di utenti *u* e *v* tali che (u, v) è un arco, è assunto che le loro forti relazioni siano proporzionali alla frazione dei progetti a cui essi hanno lavorato congiuntamente in passato. Se N_u e N_v denotano l'insieme dei progetti nei quali *u* e *v* sono noti come contributori, la loro relazione forte è definita come $\sigma_{u,v} = \frac{|N_u \cap N_v|}{|N_u \cup N_v|}$.

In fine, la funzione $w(u,v) = 100 \bullet (1 - \sigma_{u,v})$ è usata per stimare il costo di collaborazione tra *u* e *v*.

Item. Gli item nel dataset D1 corrispondono al linguaggio di programmazione richiesto dal progetto e ce ne sono 52 nel percorso considerato. Per ogni utente, è costruito il proprio set di item come collezione dei linguaggi di programmazione richiesti dai suoi progetti.

Task in GitHub. Il task corrisponde ai requisiti del progetto creato dagli utenti. I progetti in D2 e i loro requisiti (in termini di linguaggi di programmazione) sono utilizzati come task per gli esperimenti. D2 non è usato per stimare altri parametri dal momento che esso può influenzare l'esito dell'esperimento.

In aggiunta, vengono generati, artificialmente, ulteriori requisiti di progetto. Purtroppo, campionare in modo randomizzato gli item non garantisce requisiti realistici per il progetto. Per tale motivo vi è bisogno di generare requisiti sintetici.

Capacità. Per ottenere comprensione delle capacità degli utenti, è stato scelto un set di circa 2000 user divenuti attivi e vengono analizzati i commit logs relativi ai loro progetti, su di un periodo di un mese. Tra i progetti listati, è stabilito che un progetto è attivo per quell'user, se esegue almeno una commessa nell'arco di un mese. Il risultato di questo esperimento è riportato in Figura18.

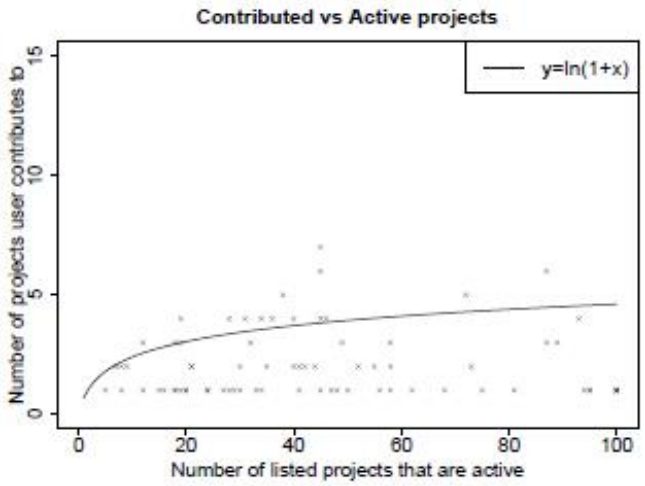

Figura 18 : Numero dei progetti a cui contribuisce un utente vs numero dei progetti listati (per 2000 utenti).

Il grafico rivela un trend interessante: il numero dei progetti a cui un utente contribuisce è spesso significativamente minore rispetto al numero dei progetti listati per l'utente. Crediamo fortemente che questa differenza sia causata dalle capacità limitate degli utenti. In fine è calcolata la capacità dell'utente u come $\ln(1 + N_u)$, dove N_u è il numero dei progetti a cui u contribuisce, come listato da GitHub.

Risultati. Il primo set di esperimenti mette in risalto il fatto che le tecniche possono significativamente migliorare il costo di collaborazione sociale nei progetti esistenti in GitHub. Per realizzare questo, si è lavorato con i progetti in D2 ed effettuato la selezione tenendo in cosiderazione i seguenti parametri: numero di utenti che contribuiscono ai progetti, linguaggi di programmazione, quanti utenti necessitano di ogni linguaggio etc.. Per ogni progetto, sono state calcolate le soluzioni restituite dagli algoritmi *MinDiamSol* e *MinAggrSol* ed è stato comparato il loro costo con quelli relativi allo steiner e al diametro. Per quasi il 40% dei progetti, è possibile osservare una riduzione del costo tra il 19% - 35%. Nella Figura 19, riportiamo il risultato per il *MinAggrSol*.

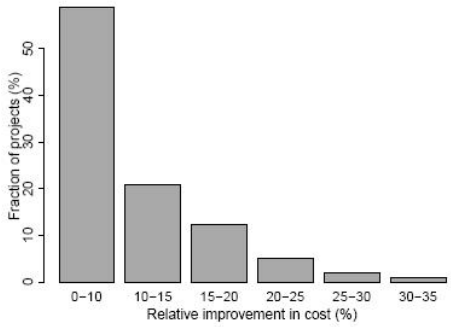

Figura 19 : Miglioramento relativo al costo di collaborazione secondo l'algoritmo MinAggrSol sui team in GitHub

Per approssimazione sul 5% dei progetti, si osserva che l'algoritmo considerato ha ottenuto una soluzione che ha costo più alto rispetto ai team attuali.
In Figura 20 è riportato il tempo di computazione dei tre diversi algoritmi al variare della dimensione della rete.

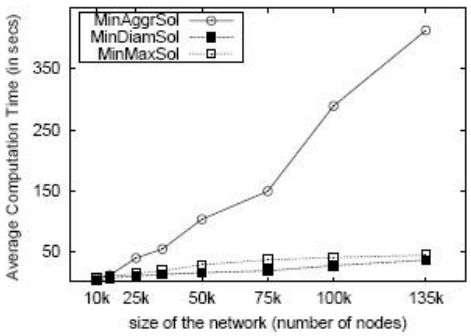

Figura 20 : Tempo di computazione dei diversi algoritmi al variare della taglia della rete.

3.8.5.2 Confronto con Baseline

Algoritmo Baseline. Per la comparazione con baseline, è implementata una versione adattata degli algoritmi *RarestFirst* e *EnhancedSteiner*, dal lavoro di Lappas.
Per un dato task, questi algoritmi trovano un team di esperti da un social network ma non sono progettati per trattare i vincoli di capacità sugli utenti. Questi algoritmi vengono modificati come segue.
Innanzitutto, è eseguito l'algoritmo ed è esaminata la sua soluzione. Se la soluzione è fattibile, l'algoritmo è dichiarato valido. Altrimenti, per ogni utente u, il cui vincolo di capacità ancora non è raggiunto, è eseguita una riallocazione degli item. Viene effettuato l'ordinamento degli item in base alla loro "rarefazione" ed essi vengono assegnati ad u (minimo incremento sul costo di copertura dell'item da altri utenti che non sono inclusi nella soluzione corrente). Successivamente i più rari b_u item vengono assegnati ad u con tecnica greedy. Dopo aver eseguito la riallocazione per tutti gli user che non rispettano il vincolo, il set degli user viene contratto in un singolo nodo del grafo e viene ripetuto lo step, fino ad ottenere una soluzione fattibile. Questi algoritmi sono denominati rispettivamente *GreedyDiam* e *GreedySteiner*.
Ottimizzazione. Attraverso alcuni iniziali insiemi di esperimenti, è osservato che *MinDiamSol* è efficiente in termini di tempo di esecuzione. D'altra parte, l'algoritmo *MinAggrSol* chiaramente non scala in proporzione ai dati; infatti occorre un tempo significativo per calcolare le distanze dei più corti percorsi tra i nodi del grafo (step 4 e 10 in *MinAggrSol*). E' stato implementato perciò uno schema di distanza che fornisce una rapida stima delle interrogazioni dello shortest path.
E' stata presa in prestito l'idea della precomputazione approssimata delle distanze di shortest path dal lavoro di Das Sharma (2010). L'approccio è quello di mantenere una struttura schematica che consiste di un piccolo numero di nodi di riferimento e di precomputare il più corto percorso delle distanze attraverso tale struttura. Gli esperimenti suggeriscono che dal salvataggio di soli 5 schemi, si può ridurre l'errore medio relativo sotto il 10% (Tabella 7).

#sketches	error	size	computation time
1	51%	10MB	5 hours
3	18.7%	32MB	18 hours
5	9.3%	55MB	28 hours

Tabella 8 : Esempio di distanze

Risultati. Gli esperimenti valutano la performance del *MinDiamSol* e *MinAggrSol* rispetto all'euristica baseline. Dapprima è determinato il valore minimo di h (lunghezza del salto massimo), per il quale l'input prende una soluzione fattibile. In Figura 21, mostriamo il risultato di questo esperimento.

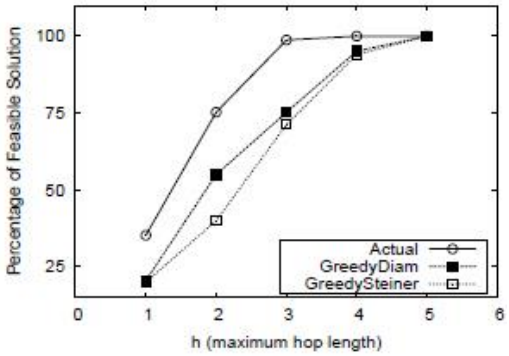

Figura 21: Risultati sperimentali per il dataset GitHub.

Come si può vedere, per un valore di *h*=3, i task (sia reali che sintetici) hanno soluzioni fattibili. Comunque, risulta che sia *GreedyDiam* sia *Greedy Steiner* falliscono per trovare soluzioni fattibili per tutti i task, a meno che non incrementiamo *h* a 5. La figura 21 riassume il risultato dell'esperimento con task reali. Gli esperimenti rivelano che *MinDiamSol* raggiunge almeno 25% di riduzione sul costo se comparato a GreedyDiam. Il miglioramento è ancor più pronunciato nella versione dello Steiner cost in cui l'algoritmo è significativamente più performante rispetto all'euristica GreedySteiner di almeno il 40%.
L'attenta analisi degli step del GreedySteiner rivela che se ignoriamo la capacità degli utenti di prendere decisioni, l'algoritmo include moltissimi nodi steiner nella soluzione e perciò, incrementa il costo. Il grafico del tempo di esecuzione indica che gli algoritmi scalano linearmente con il numero degli item in un task. La figura 22 mostra il risultato dell'esperimento con task artificiali. Per il data-set sintetico, gli algoritmi selezionati realizzano il 30% di riduzione del costo. La riduzione è irrilevante sul fattore di performance e ciò può essere attribuito al fatto che una grande frazione dei task sintetici contengono solo 3 item. Come mostrato in Figura 22, per i task con solo 3 skill, sia il baseline che gli altri algoritmi hanno una performance comparabile e il miglioramento aumenta progressivamente per grandi task.
Per un dato task, può succedere che non c'è nessun team fattibile di user che definisce un sottografo connesso. Anche se esiste qualche team, potrebbe essere possibile che gli algoritmi falliscano per trovarlo. Nonostante i team disconnessi non sono utili alla collaborazione, è interessante valutare quanto gli algoritmi li prendano in considerazione. In Figura 22 è possibile osservare la frazione dei team disconnessi restituiti dagli algoritmi e dal baseline. Il 18% dei team riportati dal *MinDiamSol* sono disconnessi. Questi sono i task per i quali non può esserci alcun team connesso nel grafo d'input. Sia MinDiamSol e GreedyDiam trovano approssimativamente la stessa frazione dei team disconnessi. GreedySteiner, d'altra parte, ritorna team disconnessi sostanzialmente più frequentemente che MinAggrSol.

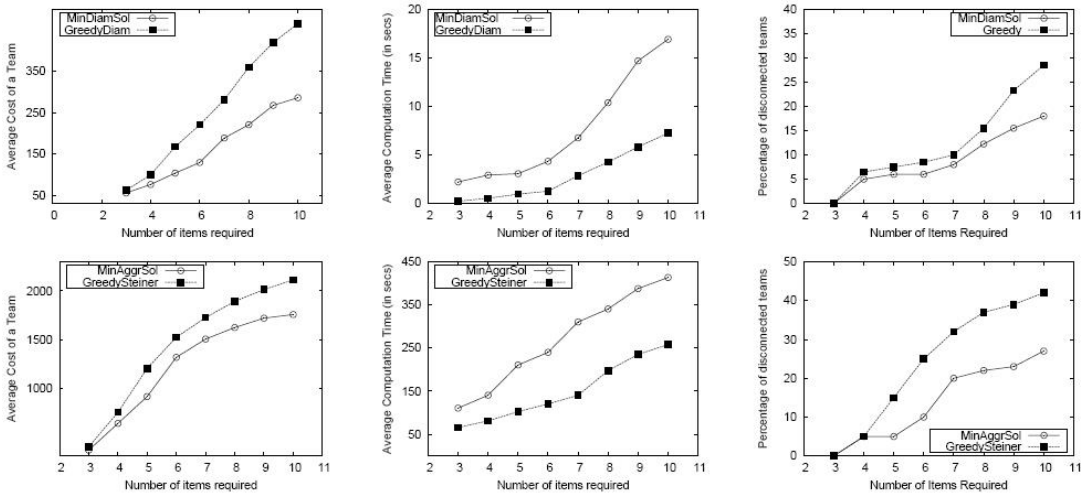

Figura 22 : Risultati sperimentali sul dataset GitHub con task sintetici.

3.8.5.3 Il dataset DBLP

Per dimostrare la generalità dell'approccio proposto da Datta et all., è stato effettuato anche un set addizionale di esperimenti su un DBLP data set.

Il secondo data-set, estratto da DBLP, consiste di una istantanea delle pubblicazioni informatiche prese a partire dal 5 Dicembre 2011. Ogni pubblicazione è scritta da un insieme di autori. Sono eseguiti gli stessi step del pre-processing come descritti da Lappas per generare i parametri di input per gli algoritmi considerati. In aggiunta, la capacità di un autore *u* è definita come la media del numero di pubblicazioni in cui *u* è stato coautore nell'anno.

Diversamente da Gitub, il dataset DBLP non ha una esplicita nozione di grafo sociale. Il social network basato sulla co-autoring è generato come segue: due autori sono connessi da un arco se essi hanno collaborato ad almeno due articoli unitamente. Il costo di collaborazione di un arco (*u*,*v*) è stimato nel seguente modo: se S_u (o S_v) denota l'insieme degli articoli di ricerca in cui *u* è co-autore (risp. *v*) allora il costo dell'arco (*u*,*v*) è stimato come $100 \cdot (1 - \frac{|S_u \cap S_v|}{|S_u \cup S_v|})$. Ancora una volta, la costante 100 è arbitraria ed è usata per evitare ogni errore di rounding nella computazione. Seguendo gli step di pre-processing, il dataset DBLP contiene 7332 autori, 19248 articoli e 2763 item distinti.

Il risultato dell'esperimento è riassunto in Figura 23.

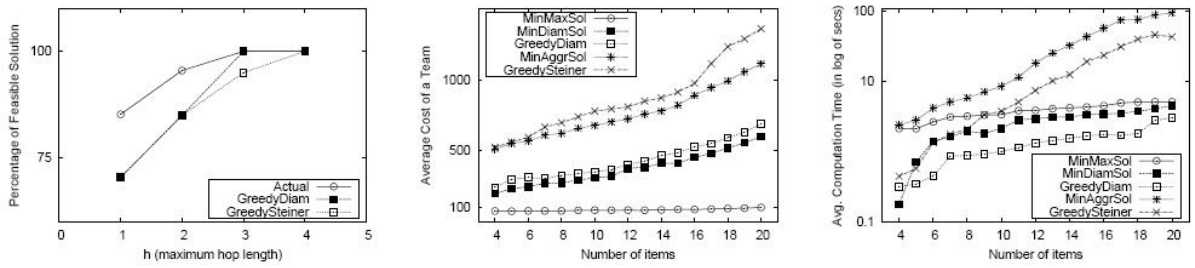

Figura 23 : Risultati sperimentali per il dataset DBLP.

Visto che il network DBLP è leggermente più denso rispetto a GitHub, gli algoritmi baseline possono trovare team fattibili con 4 salti dallo user root. Perciò, per lo scopo di questo esperimento, *h* è settato a 4. Sia *MinDiamSol* che *MinAggrSol* raggiungono circa il 20% di miglioramento rispetto

all’euristica baseline. Come menzionato prima, la costruzione sintetica della rete DBLP è leggermente più densa rispetto a GitHub e contiene utenti con molte pubblicazioni (quindi grande capacità). Questo fatto ha portato ad una leggera riduzione dei miglioramenti riportati dagli algoritmi considerati, se comparati agli esperimenti su GitHub.
Per concludere, attraverso un insieme di esperimenti discretamente grande su un dataset real-world, è stata valutata la performance degli algoritmi di team formation. I risultati sperimentali suggeriscono che gli algoritmi considerati hanno una prestazione migliore rispetto alle altre strategie naive di un significativo margine (20-40%) e scalano bene con centinaia di migliaia di utenti.

4. Scenario di riferimento

Come già chiarito, l'obiettivo della tesi è quello di introdurre un modello di ausilio al team formation per la risoluzione dei malfunzionamenti informatici in una piccola realtà dipartimentale, ponendo l'attenzione ai benefici che l'utilizzo di questo specifico strumento può apportare anche in realtà più grandi, con l'intento, casomai, di coinvolgere interi dipartimenti del Ministero presso cui opero e con l'ambizione che l'idea venga adottata anche da ambienti esterni ad esso. Vediamo in dettaglio come è strutturato l'ambiente di lavoro e quindi lo scenario in cui applicare poi il modello risolutivo, quali le procedure attuali e quali i processi migliorabili.

4.1 Ambiente di lavoro

L'*Albo Nazionale dei Segretari Comunali e Provinciali*, afferente al Dipartimento degli Affari Interni e Territoriali del Ministero dell'Interno, è una Direzione Centrale atta a recepire le Direttive Ministeriali circa la gestione di particolari figure professionali quali i Segretari degli Enti Locali (reclutamento, formazione, carriere, retribuzioni, assegnazioni, trasferimenti, pensionamenti…).

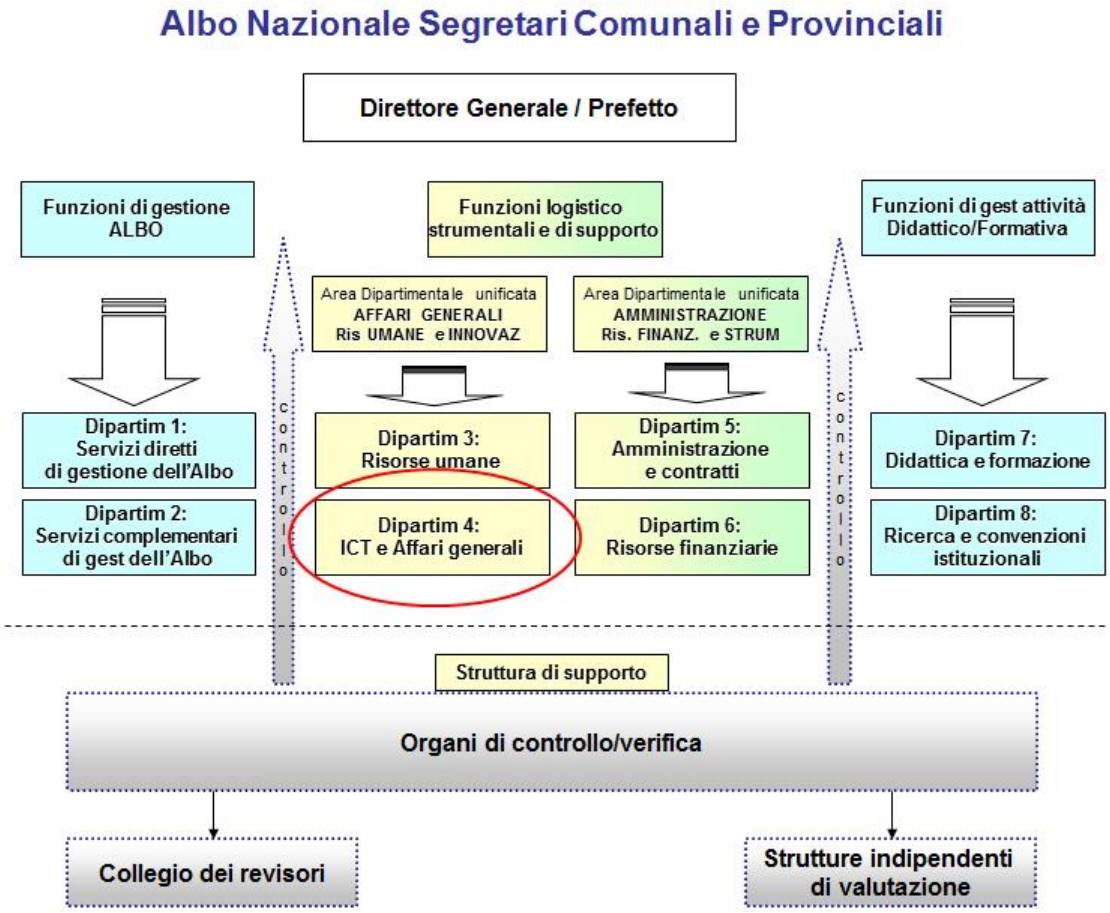

Figura 24. L'Ufficio Informatico afferisce al Dipartimento 4: ICT e AAGG

L'Ufficio Informatico di cui sono responsabile, oltre alla coordinazione di un centro di costo che gestisco personalmente, nasce soprattutto con lo scopo di recepire gli aspetti tecnologici delle eventuali modifiche alla disciplina dei Segretari, nonché per risolvere guasti hw/sw alle

apparecchiature del CED oltre che alle postazioni di lavoro dei colleghi, presenti nella sede centrale di Roma come pure dislocate nella varie sezioni territoriali.
Quest'ultimo punto richiede l'intervento dello staff afferente al mio ufficio in termini di risorse umane che di volta in volta dovranno essere assegnate ai guasti informatici. Tale staff è costituito da 3 colleghi incardinati all'interno dell'organigramma ufficiale (funzionari e operatori) e da una decina di risorse, dipendenti di aziende appaltatrici del suddetto Ente.

4.1.1 Piattaforma di help desk

La parte riguardante la ricezione degli avvisi di guasti è stata già informatizzata mediante l'utilizzo di una piattaforma di HelpDesk. L'applicativo è di tipo web based e completamente realizzato con tecnologie open source. Questa caratteristica ne permette l'integrazione in modo estremamente rapido all'interno di qualsiasi infrastruttura IT. Inoltre, richiedendo poche risorse di sistema, ne consente l'utilizzo anche su macchine server poco performanti.

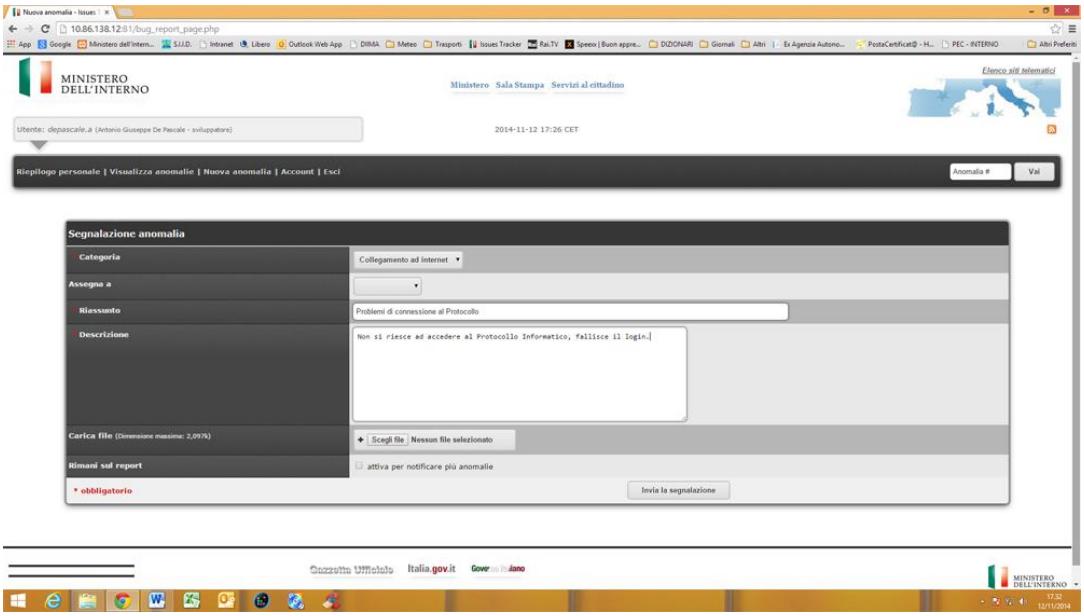

Figura 25: Piattaforma di HelpDesk lato utente – Apertura tiket

Grazie ad essa gli utenti in difficoltà (colleghi che usano apparecchiature e sistemi informatici) contattano l'Ufficio succitato "aprendo" un ticket. Il ticket, che perviene all'Ufficio Informatico sotto forma di mail, oltre ai dati del mittente, contiene vari campi descrittivi. In esso vengono specificati sede, macchina e tipologia di guasto. Inoltre il ticket è corredato già da un soggettivo livello di urgenza/priorità, assegnato dall'utente.
Tale strumento consente in realtà di personalizzare i workflow operativi (ossia lo stato che un ticket può assumere e chi può fare cosa).
La piattaforma consente lo storaggio degli interventi e dei metadati collegati ad essi (tempo impiegato alla risoluzione, risorse impegnate, tipologie di guasti…). L'estrapolazione di tali dati sotto forma tabellare e grafica consente di effettuare analisi e di aggiornare i profili dei collaboratori (fig. 27). Inoltre il tool consente l'integrazione di plugin esterni.

4.2 Procedura di assegnazione e inquadramento risorse

Addentrandoci nello scenario di riferimento, nel momento in cui sopraggiunge un malfunzionamento (o la necessità di qualsiasi altra attività informatica), l'utente compila il proprio ticket e lo invia al servizio informatico. Il ticket, sotto forma di e-mail, giunge al server di posta che lo smista nelle caselle dei gestori della piattaforma di Help Desk, secondo il workfolw illustrato in Figura 26. Giunta la mail, prima di assegnare la/le risorsa/e al guasto, vengono vagliati una serie di parametri. La prima operazione è quella di verificare se il livello di urgenza segnalato dall'utente corrisponde alla realtà. In genere viene data priorità ai guasti "bloccanti" e a quelli che inficiano l'operato dei dirigenti. A parità di urgenza si da priorità ai ticket giunti prima. Scelto il ticket da risolvere si esaminano le risorse: inizia la vera e propria procedura di *team formation*!

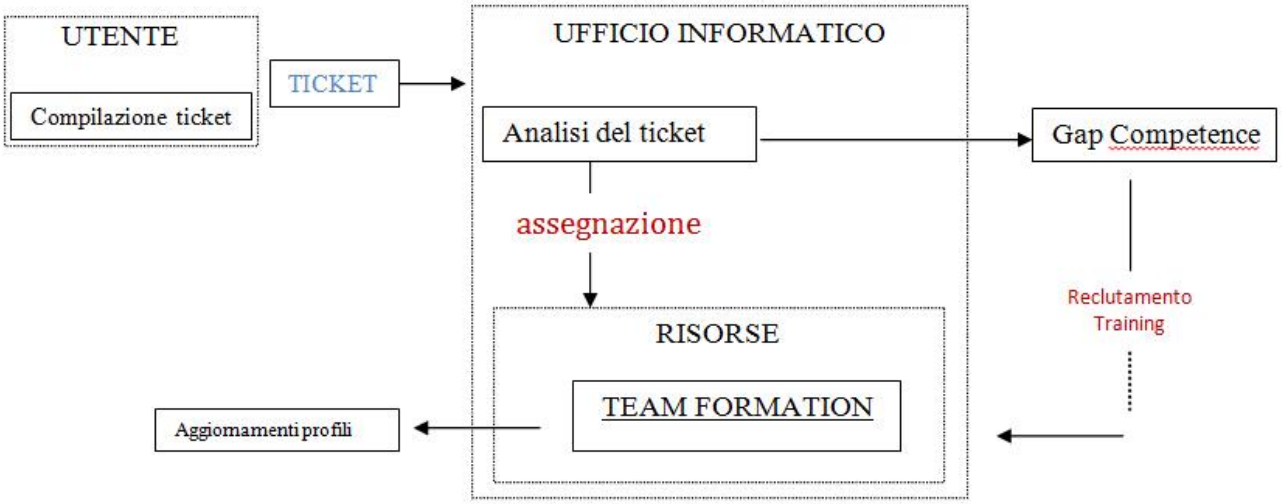

Figura 26. Attuale generico processo di gestione del ticket

Al task da risolvere verrà dato un peso crescente in base al suo grado di difficoltà ossia alla specificità delle capacità desiderate.
In particolare, soffermandoci sul blocco centrale di fig. 26, l'output dell'analisi dei ticket è in realtà una *ranked list* di task. Poi c'è una fase di analisi delle risorse con una biforcazione che porta ad un blocco di team formation (assegnazione delle risorse ai task) e ad un blocco di reclutamento.
In particolare, se le competenze richieste non esistono nel contesto attuale, si dovrà pensare ad una fase di recruitment, analizzando se conviene formare una risorsa presente o reclutarne una ex novo (gap competence).
Si filtrano quindi solo le risorse realmente presenti, e tra queste, quelle libere da altre risoluzioni. Tra quelle "disponibili", in genere scelgo quelle più performanti, pur sapendo che questa tipologia di scelta potrebbe comportarmi problemi sul lungo periodo.
Assegnato il gruppo di lavoro (o il singolo worker) al task effettuo delle verifiche per controllare che il guasto sia risolto in tempi ragionevoli, aggiornando alla fine le miei personali informazioni reputazionali sulle risorse. Per la computazione dei tempi di risoluzione utilizzo uno strumento messo a disposizione dalla piattaforma di Helpdesk prima illustrata che mi restituisce i timesheet dei collaboratori.

AGES-SSPAL: Report annuale status				
x dot				
2241 Items.				
30 Oct, 2012				
Ticket #	Summary	Days Open	Time in Minutes	Total Ticket Cost
1	Welcome to the Spiceworks Help Desk	12 days	0	€0,00
4	Watch the Help Desk Overview Video	12 days	0	€0,00
5	connettività	< 1 day	15	€20,00
6	non parte il PC	< 1 day	5	€6,67
7	mappatura disco di rete	< 1 day	60	€80,00
8	installazione stampante di rete	< 1 day	10	€13,33
9	Stratos	< 1 day	15	€20,00
10	Digital	< 1 day	15	€20,00
11	installazione stampante di rete	< 1 day	45	€60,00
12	IP fisso	< 1 day	5	€6,67
15	Installazione Flash e Shockqwave player	< 1 day	5	€6,67
16	test portale pubblico	< 1 day	0	€0,00
17	backup	< 1 day	0	€0,00
18	test ticket	< 1 day	0	€0,00
19	test	< 1 day	0	€0,00
20	Link a sito AGES Lazio	< 1 day	15	€20,00
21	installazione stampante	7 days	30	€40,00
22	test impostazioni mail di sistema	< 1 day	0	€0,00
28	installazione licenze ADOS	< 1 day	30	€40,00
29	richiesta installazione webcam	< 1 day	60	€80,00
30	installazione licenze ADOS	< 1 day	30	€40,00
31	Tastiera non funzionante	< 1 day	10	€13,33
32	problemi posta elettronica	< 1 day	30	€40,00
33	problemi posta elettronica	< 1 day	10	€13,33
34	problemi posta elettronica	< 1 day	20	€26,67
47	Sostituzione nastri backup	< 1 day	15	€20,00
48	installazione stampante di rete	< 1 day	15	€20,00
49	Problemi con sito	< 1 day	10	€13,33
50	sostituzione nastri backup	< 1 day	15	€20,00
51	Problemi invio posta	< 1 day	120	€160,00
52	installazione driver stampante	< 1 day	30	€40,00
53	reset password Unicredit	< 1 day	15	€20,00

Figura 27. Strumenti a disposizione del decision maker: timesheet e report.

Il mio Ufficio si occupa anche dello sviluppo di veri e propri progetti di manutenzione, modifica o creazione di semplici software o infrastrutture, nonché l'istruzione di atti amministrativi riguardanti la parte informatica; anche questi più complessi interventi, vengono tracciati tramite l'apertura di un ticket alla pari di un "guasto" informatico. Le risorse anche in questi casi, sono assegnate con lo stesso criterio, tenendo presente il concetto di multi-progetto. Ossia, il team incaricato di condurre il progetto vi lavora quando le risorse ad esso afferenti non sono impegnate nella risoluzione dei ticket provenienti dalla piattaforma di helpdesk. Da questo punto di vista si portano avanti più attività in simultanea e le stesse risorse sono assegnate a più attività.
Le collaborazioni di cui si avvale il mio Ufficio attualmente sono di due tipologie: contrattualizzate ed esterne. Le prime risultano incardinate nei ruoli ministeriali e principalmente danno l'indirizzo alla risoluzione dei guasti; le figure esterne invece provengono da ditte appaltatrici che forniscono figure professionali più orientate alla fase operativa. Ogni risorsa è inquadrata in un preciso contesto lavorativo ed è qualificata mediante il proprio curriculum. Ad esempio la risorsa che si occupa della gestione del Database Oracle dove sono memorizzate tutte le informazioni riguardanti le carriere dei Segretari, è specializzata in Db ed il suo curriculum è caratterizzato da certificazioni Oracle con le specifiche sulle altre collaborazioni di quel tipo. Man mano che crescono o si modificano gli skill dei collaboratori mi occupo personalmente di aggiornare i loro profili con l'aggiunta delle certificazioni conseguite o dei corsi a cui hanno partecipato. Altre informazioni relative alle risorse umane sono quelle circa gli anni di esperienza sulle varie discipline conosciute. Anche se nel pubblico impiego si sta cercando di implementare un modello comune per storare gli skill dei vari dipendenti (frutto di alcuni metodi di valutazione analitica delle prestazioni - *repertory grid*), ad ora le info relative ai miei collaboratori sono da me tenute in una sorta di fascicolo del personale informatico, utilizzato al momento di aggiornamenti ed assegnazioni. Le particolari informazioni circa le collaborazioni pregresse attualmente non sono sfruttate ai fini dell'assegnazione. L'unico fattore che, ad ora, si è cercato di tener presente è quello di evitare la stessa assegnazione collaborativa contemporaneamente a due persone che in passato hanno avuto frizioni.

4.3 Formalizzazione del modello attuale

Riassumendo, il processo che attualmente vede coinvolte le risorse umane afferenti all'Ufficio Informatico, dedicato alla risoluzione dei malfunzionamenti, può essere formalizzato come di seguito descritto.

L'insieme R = $\{r_1, \ldots, r_m\}$ è costituito dalle risorse a disposizione, ognuna skillata in una o più delle *m* discipline; l'insieme T = $\{t_1, \ldots, t_n\}$ racchiude gli *n* task da risolvere. L'azione che effettua il supervisore nell'assegnare una competenza ad un profilo è tradotta dalla funzione *skill*(*r*), che restituisce l'insieme $\{c_{p1}, \ldots, c_{pn}\}$, dove c_{pi} è l'indice corrispondente ad una competenza posseduta dalla risorsa *r*. L'abilità totale richiesta ossia l'insieme di tutte le competenze richieste dai vari task in atto di elaborazione è pari ad A = $\{A_{t_1}, \ldots, A_{t_n}\}$. In cui A_{t_1} sarà l'insieme di tutte le competenze c_p richieste per il task t_1. La variabile decisionale *x*, che stabilisce se una risorsa abbia capacità necessarie o meno, è settata nel seguente modo:

$$x(r,t)=\begin{cases}1 \text{ se la risorsa } r \text{ "è abile a coprire" il task } t\ ;\\ 0 \text{ se la risorsa } r \text{ non ha capacità sufficienti per il task } t\ .\end{cases}$$

Funzione Obiettivo. Il processo attualmente si limita a coprire i ticket aperti impiegando il minor numero possibile di risorse ovvero di minimizzare il numero di task pro capite per ottimizzare la performance complessiva. La funzione obiettivo quindi sarà della forma:

$$Assegnamento = min\ (assegnazioni\ \forall r)$$

vincolata da:

$$\forall t \epsilon T \sum_{r \epsilon R} x(r,t) \geq 1 \qquad \text{(v4.1)}$$

$$\bigcup_{r \epsilon R} skill(r) \supseteq \bigcup A_t \quad \forall t \text{ ancora attivo} \qquad \text{(v4.2)}$$

$$\bigcup_{r \epsilon R} \bigcup_{i=1}^{d(r)} skill(r_i) - \bigcup_{i=1}^{n-1} A_{t_i} \supseteq A_{t_n} \qquad \text{(v4.3)}$$

$$\forall t \epsilon T \sum_{r \epsilon R^*} x(r,t) \leq 1 \qquad \text{(v4.4)}$$

Il vincolo (v4.1) garantisce che ci sia almeno una risorsa utile al task t. Se questo vincolo non è soddisfatto vorrà dire che non ho le risorse a disposizione in grado di risolvere l'attività. Bisognerà attivare quindi una strategia di reclutamento o di training (fase di *Gap Competence* di figura 26).

Il vincolo (v4.2) stabilisce che l'insieme delle abilità complessive disponibili includa almeno l'insieme dell'abilità totale richiesta. Il responsabile fissa a priori sia i valori dei parametri *skill*(*r*) in base ai curriculum ed alle certificazioni, sia il contenuto dei vari insiemi $A_{t_1}, \ldots, A_{t_n}$, l'unione dei quali comporrà l'insieme complessivo delle competenze richieste A (un esempio è mostrato in tabella 9). Il vincolo (v4.2) mi consente di esser sicuro che, pur impegnando risorse attualmente attive in altri task, riuscirò prima o poi a risolvere il task appena giunto.

La variabile *n*, che tiene il conto dei task giunti, sarà presente nella relazione che limita superiormente il numero di assegnazioni pro capite. Il parametro *d(r)* introdotto nella (v4.3) dà flessibilità al vincolo consentendo di personalizzare il numero massimo di assegnazioni per ogni utente.

Il vincolo (4.2) volutamente non tiene conto delle assegnazioni multiple, tendendo a minimizzare le stesse per risorsa. Se in prima battuta tale vincolo non è soddisfatto, con il vincolo 4.3 sarà possibile, successivamente, esaminare la possibilità di multi-assegnazione sulle risorse, tenendo conto del grado di assegnazioni pro-capite *d(r)*, stabilito dal responsabile.
In fine, il vincolo di natura interpersonale (v4.4), è legato a fattori caratteriali o alle vicende passate dei collaboratori. Questo è particolarmente semplice da esprimere: è sufficiente imporre che la somma delle variabili *x(r,t)* su *r* nel sottoinsieme R^*, formato dalle persone che tra loro hanno attriti, sia al più uno per ogni task $t \in T$.

4.4 Caso d'uso

Vediamo un esempio esplicativo di quello che accade tutti i giorni nel contesto sopra descritto.
Riduciamo al minimo le risorse e i task per meglio comprendere il flusso logico del modello attuale: considereremo 2 risorse e l'arrivo di 4 task. Inoltre per stressare maggiormente il modello, considereremo i task consecutivi e paralleli, nel senso che all'arrivo del task i-esimo i ticket relativi ai precedenti i-1 task sono ancora tutti aperti. Inoltre stabiliamo che nella compilazione del ticket l'utente possa marcare il guasto come "urgente" se bloccante, o "relativo" se non urgente. A parità di parametri, i guasti del primo tipo hanno priorità (cap. 4.2).
E' stata stabilita a priori una tabella delle attività che il mio team è capace di svolgere:

Identificativo attività	Tipologia di attività
c_{p1}	Manutenzione ordinaria
c_{p2}	Installazione Hardware
c_{p3}	Installazione Software
c_{p4}	Gestione sito web
c_{p5}	Aggiornamento Db
c_{p6}	Implementazione sito web
c_{p7}	Implementazione nuovo Sw

Tabella 9. Esempio di tabella delle competenze/attività. Ogni risorsa *r* possiederà una o più competenze c_p. Ad ogni task *t* saranno associate delle attività A_t richiedenti competenze c_p.

Ipotizziamo che, in una precisa giornata lavorativa, abbia due risorse a disposizione: r_1 (bravo ad effettuare manutenzione ordinaria e ad installare hardware) ed r_2 (bravo nell'installazione dell'hardware e software, inoltre è skillato nelle operazioni di gestione del sito web). Dall'analisi dei loro fascicoli, tramite la funzione *skill(r)*, posso effettuare l'assegnazione dei livelli di competenza come segue:
$skill(r_1) = \{c_{p1}, c_{p2}\}$
$skill(r_2) = \{c_{p2}, c_{p3}, c_{p4}\}$
Ipotizzo inoltre che tra le due risorse, in passato, non ci siano mai stati attriti, ossia $r_1 \notin R^*$ ed $r_2 \notin R^*$ (quindi il vincolo (v4.4), in questo esempio, è sempre soddisfatto).
Inoltre voglio che ad r_1 non sia assegnato mai più di un lavoro per volta $d(r_1) = 1$, mentre reputo che r_2 sia in grado di svolgere anche due compiti in simultanea $d(r_2) = 2$.

Vediamo con la gestione di 4 task paralleli come opera il modello attuale:
1. Giunge il primo ticket: *"Richiesta installazione stampante ed aggiornamento driver"*.
   Costruisco il task t_1, assegnandogli le seguenti abilità richieste $A_{t_1} = \{c_{p1}, c_{p2}\}$.

$x(r_1, t_1)$ = 1 perché nello skill di r_1 ci sono le competenze $\{c_{p1}, c_{p2}\}$;
$x(r_2, t_1)$ = 0 perché r_2 non ha le competenze richieste.
Esaminiamo i vincoli:

- (v 4.1) 1+0 ≥ 1 SODDISFATTO
- (v 4.2) $skill(r_1) \cup skill(r_2) \supseteq A_{t_1}$
  $\{c_{p1}, c_{p2}\} \cup \{c_{p2}, c_{p3}, c_{p4}\} \supseteq \{c_{p1}, c_{p2}\}$
  $\{c_{p1}, c_{p2}, c_{p2}, c_{p3}, c_{p4}\} \supseteq \{c_{p1}, c_{p2}\}$ SODDISFATTO
- (v 4.3) $\{c_{p1}, c_{p2}\} \cup (\bigcup_{i=1}^{2}\{c_{p2}, c_{p3}, c_{p4}\}) - \{\emptyset\} \supseteq \{c_{p1}, c_{p2}\}$ SODDISFATTO

Possiamo assegnare la risorsa r_1 al task t_1 ossia $r_2 \leftarrow t_1$.
Osservazione: il modello attuale tenta di assegnare tutte le attività inerenti lo stesso task ad un'unica risorsa, riducendo il concetto di "team" solo ai casi in cui non esiste alcuna risorsa le cui competenze coprano totalmente quelle richieste!

2. Giunge il secondo ticket: *"Non riesco ad inserire una news sulla home page del sito web istituzionale per mancanza di un software sulla mia postazione."*
Costruisco il task t_2, assegnandogli le seguenti abilità richieste $A_{t_2} = \{c_{p3}, c_{p4}\}$.
$x(r_1, t_2)$ = 0 perché r_1 non ha le competenze richieste;
$x(r_2, t_2)$ = 1 perché nello skill di r_2 ci sono anche le competenze $\{c_{p3}, c_{p4}\}$.
Esaminiamo i vincoli:

- (v 4.1) 0+1 ≥ 1 SODDISFATTO
- (v 4.2) $skill(r_1) \cup skill(r_2) \supseteq A_{t_1} \cup A_{t_2}$ // perché il ticket t_1 è ancora aperto
  $\{c_{p1}, c_{p2}\} \cup \{c_{p2}, c_{p3}, c_{p4}\} \supseteq \{c_{p1}, c_{p2}\} \cup \{c_{p3}, c_{p4}\}$
  $\{c_{p1}, c_{p2}, c_{p2}, c_{p3}, c_{p4}\} \supseteq \{c_{p1}, c_{p2}, c_{p3}, c_{p4}\}$ SODDISFATTO
- (v 4.3) $\{c_{p1}, c_{p2}\} \cup (\bigcup_{i=1}^{2}\{c_{p2}, c_{p3}, c_{p4}\}) - \{c_{p1}, c_{p2}\} \supseteq \{c_{p3}, c_{p4}\}$
  SODDISFATTO

Possiamo assegnare la risorsa r_2 al task t_2 ossia $r_2 \leftarrow t_2$.

3. Giunge il terzo ticket: *"Si richiede scansione PC".*
Costruisco il task t_3, assegnandogli le seguenti competenze richieste $A_{t_3} = \{c_{p1}\}$.
$x(r_1, t_3)$ = 1 perché nello skill di r_1 ci sono anche le competenze $\{c_{p1}\}$;
$x(r_2, t_3)$ = 0 perché r_2 non ha le competenze richieste.
Esaminiamo i vincoli:

- (v 4.1) 1+0 ≥ 1 SODDISFATTO
- (v 4.2) $skill(r_1) \cup skill(r_2) \supseteq A_{t_1} \cup A_{t_2} \cup A_{t_3}$
  $\{c_{p1}, c_{p2}\} \cup \{c_{p2}, c_{p3}, c_{p4}\} \supseteq \{c_{p1}, c_{p2}\} \cup \{c_{p3}, c_{p4}\} \cup \{c_{p1}\}$
  $\{c_{p1}, c_{p2}, c_{p2}, c_{p3}, c_{p4}\} \supseteq \{c_{p1}, c_{p2}, c_{p3}, c_{p4}, c_{p1}\}$
  $\{c_{p2}\} \not\supseteq \{c_{p1}\}$ NON SODDISFATTO

Non possiamo assegnare la risorsa r_1 (unica risorsa compatibile) al task t_3, perché l'abilità c_{p1} è già in uso per risolvere un'altro task.

- (v 4.3) $\{c_{p1}, c_{p2}\} \cup (\bigcup_{i=1}^{2}\{c_{p2}, c_{p3}, c_{p4}\}) - \{c_{p1}, c_{p2}, c_{p3}, c_{p4}\} \supseteq \{c_{p1}\}$
  $\{c_{p2}, c_{p2}, c_{p3}, c_{p4}\} \not\supseteq \{c_{p1}\}$ NON SODDISFATTO

Evidentemente la risorsa r_1, l'unica in cui è presente la competenza c_{p1}, non ammette multissegnazioni, cioè *d(r)*=0. Bisogna attendere che la risorsa si liberi!

4. Giunge il quarto ticket: *"Si richiede l'installazione dello scanner."* .
   Costruisco il task t_4, assegnandogli le seguenti abilità richieste $A_{t_4} = \{c_{p2}\}$.
   $x(r_1, t_4)$ = 1 perché nello skill di r_1 ci sono anche le competenze $\{c_{p2}\}$.
   $x(r_2, t_4)$ = 1 perché nello skill di r_2 ci sono anche le competenze $\{c_{p2}\}$.
   Esaminiamo i vincoli:
   - (v 4.1) $1+1 \geq 1$ SODDISFATTO
   - (v 4.2) *skill*$(r_1) \cup$ *skill* $(r_2) \supseteq A_{t_1} \cup A_{t_2} \cup A_{t_4}$ // perché nel frattempo i ticket sono tutti aperti
     $\{c_{p1}, c_{p2}\} \cup \{c_{p2}, c_{p3}, c_{p4}\} \supseteq \{c_{p1}, c_{p2}\} \cup \{c_{p3}, c_{p4}\} \cup \{c_{p2}\}$.
     $\{c_{p1}, c_{p2}, c_{p2}, c_{p3}, c_{p4}\} \supseteq \{c_{p1}, c_{p2}, c_{p2}, c_{p3}, c_{p4}\}$ SODDISFATTO
   - (v 4.3) $\{c_{p1}, c_{p2}\} \cup (\bigcup_{i=1}^{2}\{c_{p2}, c_{p3}, c_{p4}\}) - (\{c_{p1}, c_{p2}\} \cup \{c_{p3}, c_{p4}\}) \supseteq \{c_{p2}\}$ SODDISFATTO

   Possiamo assegnare la risorsa r_2 al task t_4 ossia $r_2 \leftarrow t_4$.

4.5 Problematiche attuali e fasi migliorabili

Una caratteristica del processo sopra descritto è quella di tener presente le competenze/abilità per intero e non come parte frazionaria. Sfruttando dei livelli graduati di competenze e dei livelli di collaborazione tra le risorse sarebbe possibile assegnare il task anche se le risorse sono parzialmente occupate. Inoltre, in tale contesto, potrebbe essere d'ausilio assegnare dei livelli di difficoltà alle varie attività.
Questa fase, vedremo, è stravolta mediante l'uso dei team di lavoro che contemplano l'utilizzo del "trust". Analizzando il grado di trust tra le risorse, infatti, è facile accoppiare le stesse per farle cooperare allo stesso obiettivo.
In genere, basando la mia attuale strategia sul concetto di "risolvi prima che puoi il risolvibile", tendo a monopolizzare le risorse più brave e contemporaneamente a demotivare quelle meno skillate. Questa è una problematica che tenteremo di risolvere con il nuovo approccio "motivazionale" descritto nel capitolo successivo. In tale aspetto, si potrà considerare anche la diversa classificazione delle risorse, tra incardinate ed esterne, per valutarne la differente motivazione insita in esse.
C'è il rischio che le risorse più performanti siano iper-utilizzate: un collaboratore con *i* abilità e parametro di multiutilizzo d(*n*) rischia di avere *i·n* differenti utilizzi contemporaneamente! Questa problematica è superabile considerando il task come parametro unitario anziché come insieme di singole competenze.
Il concetto delle compatibilità tra collaboratori è speso affrontato in maniera volatile e si cerca, la dove occorrono più risorse, di assegnare quelle che hanno già in passato collaborato con esito fruttuoso. In tale ottica cercheremo di arricchire il vincolo (v4.4) con parametri che ci dicano qual è il grado ben preciso di "buona collaborazione" tra gli user.
Attualmente la Direzione Centrale presso cui opero non utilizza alcuna struttura dati particolare in cui salvare i profili dei worker, inoltre non è memorizzata nemmeno alcuna informazione relazionale tra essi. Tale aspetto è facilmente superato mediante l'utilizzo di una struttura a rete sociale, come ampiamente sperimentato nello stato dell'arte.
Il gap compentence potrà essere affrontato, successivamente, dal punto di vista delle competenze delle risorse disponibili e su come conviene coprire ruoli non presenti. Sarà possibile esaminare

quanto è utile formare le risorse interne, altre oppure quanto sarà più economico approvvigionarsi di esterni.
In fine e non per ultimo, l'analisi dei costi verrà effettuata in base al tempo impiegato dall'informatico di turno per risolvere completamente il guasto, rapportato al prezzo unitario ore/uomo.

5. Modello risolutivo

E' assodato che la motivazione sociale sul posto di lavoro incrementa lo sviluppo di fedeltà e impatta direttamente sulle performance del dipendente.
Per migliorare il modello reale, descritto nel paragrafo precedente, useremo nozioni di tipo sociale che, al meglio delle nostre conoscenze, appaiono di nuova generazione. A queste misceleremo altre parametrazioni derivanti dai modelli elencati nello stato dell'arte, o perlomeno quelle che si sono rivelate nel tempo più redditizie. Ad esempio nell'utilizzo del social network contenente la profilazione degli utenti, ripreso dal lavoro di Lappas (2009), specificheremo le azioni che caratterizzano le connessioni in tale struttura sociale.
Verrà introdotto soprattutto il concetto motivazionale del *trust* all'interno di una organizzazione.
E' possibile definire il trust come uno stato mentale che include l'aspettativa e il convincimento (Coppola, 2000).
L'idea alla base della metodologia qui introdotta è che l'analisi del trust, se adeguatamente contestualizzata, può supportare le decisioni di team formation. In particolare, nel nostro caso, ciò avviene considerando il grado di buona collaborazione combinato con le competenze più adatte per sviluppare un dato progetto, d'altronde questi principi erano già alla base della *Memoria Organizzazionale* di James P. Walsh (1991). Christianson e Harbinson (1997) affermarono che il trust è "la ferma credenza nella capacità di un'entità di comportarsi in modo affidabile, sicuro e attendibile all'interno di uno specifico contesto".
Tipicamente, nelle banche dati delle organizzazioni che si occupano di gestire le risorse sono memorizzate solo informazioni relative ai livelli di abilità, non vi è nessuna considerazione circa il concetto di trust! La mancanza di trust nelle competenze può essere identificata come una delle principali barriere alla condivisione di esperienze all'interno delle comunità on-line, scoraggiando qualsiasi forma di motivazione e partecipazione alle attività di lavoro (Ardichvili, 2003).
Considero il trust una delle principali qualità ancora poco studiate e disponibili per le aziende al fine di gestire attivamente la cooperazione e la condivisione di conoscenza. Definiremo un metodo per calcolare il grado di trust che un lavoratore ha nelle specifiche competenze di un suo collega, introducendolo nel nostro "modello risolutivo" di team formation. In particolare prenderò spunto da un approccio preesistente (nel settore del learning (Capuano, 2011)) per effettuare il calcolo del grado di trust, contestualizzandolo nell'ambito lavorativo sopra descritto.

5.1 La nostra metodologia di team formation

Nell'ambiente di lavoro descritto nel capitolo precedente è stato applicata, grossomodo, una gestione delle risorse *competency-based* che sostanzialmente utilizza soprattutto il grado di competenza delle risorse per giungere ad un buon compromesso di team formation.
Sfrutterò, da qui in avanti, l'importanza del significato di "fiducia", o come tradotto in inglese *trust*, nell'effettiva necessità prima di trasmettere la consapevolezza delle competenze e poi di confrontare le stesse.
Come visto finora, la gestione basata unicamente sulle competenze consente alle aziende di relazionare l'utilizzo delle risorse umane in base alle loro abilità al fine di plasmare la forza lavoro ed ottimizzare i risultati, ma spesso nell'applicazione al mondo reale, emergono variabili che sfuggono ai rigidi modelli sperimentati.
Questa sezione propone un approccio migliorativo alle attività di team formation, attraverso calcoli sociali sul grado di *trust-in-competencies* e la sua introduzione nel contesto prima descritto.

Il principale obiettivo di questo capitolo e pertanto quello di definire una proposta di arricchimento del modello visto in precedenza con informazioni di trust, per supportare e migliorare (considerando il trust come un modo per aumentare la qualità della condivisione delle conoscenze) le attività e gli ambienti di working collaborativo.
Il flusso relativo alla metodologia pensata è mostrato nella seguente figura.

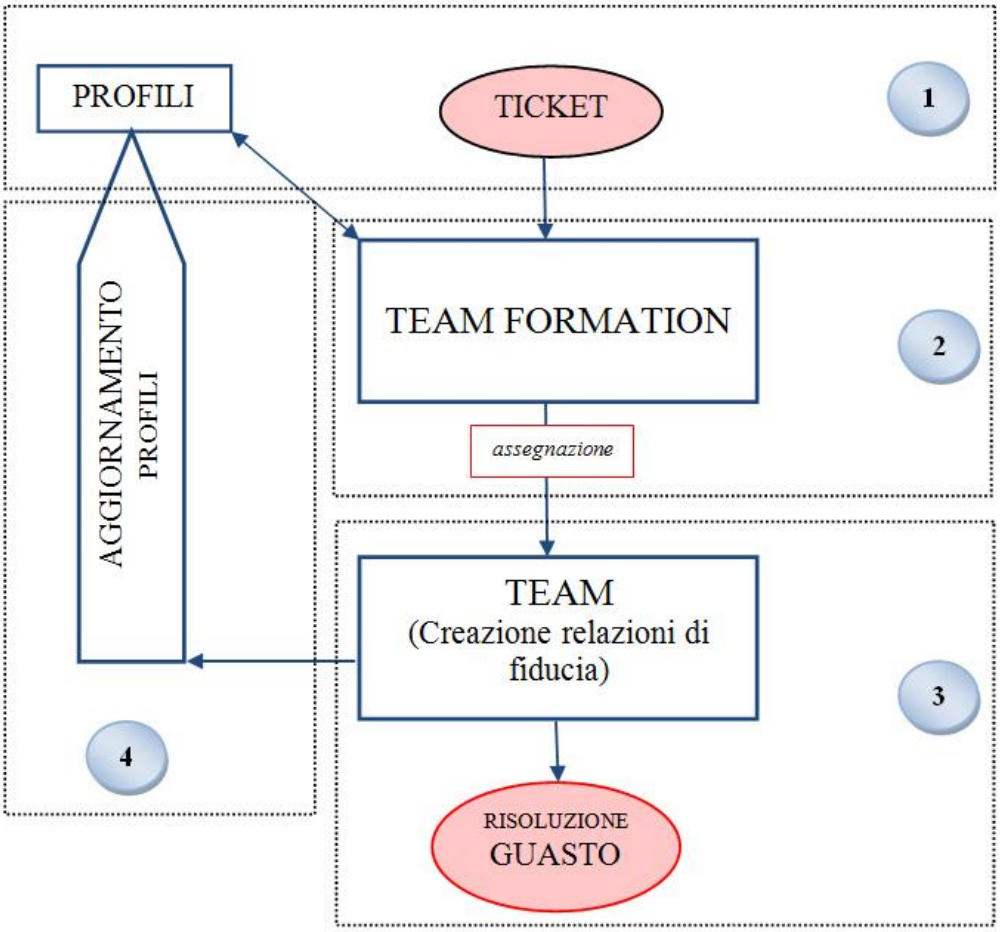

Figura 28. Metodologia di supporto al Team Formation

La metodologia consiste di 4 moduli.

1. Il primo modulo prende in ingresso il ticket, che nasce come mostrato in figura 26. L'obiettivo ricevuto in input sarà costituito da una serie di task (attività) da eseguire. Ad ogni ticket possono corrispondere una o più task. Ad essi, in base alle competenze richieste, il supervisore assegnerà un grado di difficoltà (la cui somma nel modello precedente (v4.2) era pari ad *A*). In questa fase, il supervisore ha a disposizione anche tutti i profili dei suoi collaboratori che verranno aggiornati come vedremo a breve. Egli utilizzerà nella fase successiva (2) la nostra strategia di selezione per filtrare i suddetti profili e selezionare gli skill adatti.
2. La seconda fase quindi è quella pura di team formation. In essa verrà costituito il team di lavoro che opererà secondo l'algoritmo che vedremo nel modulo 3 (Cap 5.5).
3. Nella fase 3, assegnata la squadra di lavoro (output della fase 2), viene utilizzata una rete sociale che consente, man mano che il task si risolve, di creare i gradi di trust tra gli user coinvolti nel team formation in oggetto. Questa è la fase innovativa, in cui vengono adoperati

i concetti strategici di trust nella fiducia e nelle competenze, come meticolosamente descritto nei paragrafi successivi.

4. L'ultima fase (4) è di puro aggiornamento dei profili utenti e dei gradi di trust, per successive assegnazioni.

Nei successivi paragrafi vedremo in dettaglio le 4 fasi.

5.2 Generazione dell'Input

Descriviamo in questa sezione gli input che prende il processo. La fase 1 è di concatenamento tra quella di aggiornamento dei profili (fase 4) quella pura di team formation (fase 2). Tale fase prevede la strutturazione dei profili dei dipendenti e la ricezione del ticket con la successiva classificazione del suo grado di difficoltà.

5.2.1 I profili utente

Inizieremo con l'organizzazione dei profili, descrivendo brevemente la struttura utilizzata per essi.
Utilizzeremo una semplice struttura dati in cui verranno esplicitamente trattati i profili e le competenze degli impiegati, le relazioni e le informazioni circa l'ambiente di lavoro e soprattutto la fiducia riposta nelle competenze dei colleghi. La struttura sarà interrogata per supportare la fase di making-decision nel processo di team formation.
Nel modulo 1 di figura 28 sono storati i profili dei collaboratori, organizzati in una struttura dati in cui ogni risorsa possiede una serie di valori assoluti (competenze) e ulteriori valori relativi al rapporto con le altre risorse (trust).
In tale strutture occorre chiarire per bene il concetto di competenza, definita come l'astrazione per creare poi, capacità e abilità che saranno quindi meglio gestibili ed indirizzabili. Prima di classificare e definire i rispettivi contesti, le competenze devono essere ben gestite. Il primo approccio è di dividere le competenze comportamentali da quelle tecniche.
Le ***competenze Tecniche*** sono specifiche e dipendono dalla tipologia di lavoro. Queste ultime competenze potrebbero variare, per esempio, nel settore informatico, dall'abilità per la programmazione Java alla capacità di analisi della rete.
Le ***competenze Comportamentali*** includono quei soft skills e quelle attitudini che consentono ad una persona di esplicare bene una specifica funzione, come ad esempio: comunicazione, problem-solving, deontologia, capacità di progettazione, attitudine all'organizzazione, negoziazione, etc..
Per poter modellare i profili degli impiegati e poter effettuare una buona classificazione si possono usare particolari strutture contenenti i metadati ricavati dalla carriera lavorativa, dal curriculum scolastico, dalle certificazioni che ogni impiegato ha acquisito, nonché dalle collaborazioni passate.

Categoria	Sottocategoria	Esempi	Indice
Competenze Tecniche	Conoscenze	Linguaggio Java	c_1
		Modellazione logica	c_2
		Notazioni UML	c_3
	Skill tecnici	Programmazione Java	…
		Modellazione UML	…
		Project Management	…
Competenze comportamentali	Capacità di pensiero	Problem-solving	
		Innovazione & creatività	
		Apprendimento	
	Attitudini relazionali	Lavoro di gruppo	
		Relationship management	
		Comunicazione	
		Capacità di insegnare	
	Capacità operative	Orientamento al risultato	
		Planning & organizzation	
		Decision-making	
	Qualità personali	Motivazioni	
		Responsabilità	
		Self-management	
		Integrità	

Tabella 10. Esempio di tabella relativa alle competenze (tecniche e comportamentali) di un profilo.

Vedremo poi come sarà arricchita tale astrazione, mediante le informazioni riguardanti il Trust (fase 4).

5.2.2 I task: classificazione e competenze richieste

Mediante l'utilizzo della piattaforma di Helpdesk, introdotta nel cap. 4, al responsabile giungono dei ticket, che contengono un'insieme di attività (o task) da svolgere.
Al fine di verificare se siamo in possesso delle competenze sufficienti per risolvere il ticket, è importante che ad ogni task venga assegnato un insieme di competenze dato dalla somma delle singole abilità richieste. Per far ciò il responsabile mantiene una tabella che traccia tutte le attività svolte in passato, le abilità richieste che si sono rese indispensabili per la risoluzione dei guasti e quindi la difficoltà associata ad ogni abilità (tabella 11). Inoltre ad ogni attività è associato un indice che le relaziona alle abilità presenti nella tabella 10 delle competenze.

TASK i-esimo	**Identificativo attività**	**Tipologia di attività**	**Indice**
	1.	Manutenzione ordinaria	c_{10}
	2.	Installazione Hardware	c_{10}
X	3.	Installazione Software	c_{10}
	4.	Gestione sito web	c_1
	5.	Aggiornamento Db	c_7
X	6.	Implementazione sito web	c_1
	7.	Implementazione nuovo Sw	c_1

Tabella 11. Esempio di tabella delle attività/abilità riferito al task i-simo.

$A=\{a_3, a_6\}$, con abilità richieste: c_1 , c_{10}.

In particolare l'insieme complessivo delle abilità richieste da un task è data da $A=\{a_1, a_2, \ldots, a_n\}$, dove a_i è l'attività i-esima.

5.3 Creazione delle relazioni di fiducia

Per poter comprendere a pieno l'intero processo di formation occorre conoscere nei dettagli innanzitutto cosa accade nel modulo 3 del modello schematizzato nella figura 28, quando le risorse collaborano alla risoluzione dei guasti informatici. Tale modulo, a nostro avviso, rappresenta il core di tutto il processo.
Partiamo dall'assioma che l'osservazione è il primo passo verso la formazione di una relazione di fiducia, pertanto renderemo "osservabile" e "valutabile" l'operato di una risorsa agli occhi di un'altra. Al fine di raggiungere l'obiettivo collaborativo e di completare i profili delle risorse, sarà definito un social network con caratteristiche trust-based (dove i nodi sono i dipendenti e gli archi rappresentano le relazioni tra essi). La natura sociale della rete permetterà di superare la limitazione del modello tradizionale, migliorandolo con l'utilizzo delle informazioni di fiducia (o trust). Il trust nelle competenze sarà pertanto calcolato con l'uso di un approccio sociale, cioè considerando le interazioni dirette tra due lavoratori e usando un social network per diffondere i gradi di trust. Vediamo come è strutturata questa particolare rete sociale in cui avviene lo scambio di informazioni circa il grado di trust.
L'insieme delle entità che condividono un obiettivo (la risoluzione del ticket) danno luogo ad un *processo cooperativo* ossia l'insieme delle attività svolte dalle suddette entità in un determinato ambito (o contesto). Il *contesto* è dato dal processo cooperativo più l'obiettivo. Nel nostro contesto ad una risorsa umana possono essere assegnate una o più attività (task) da condividere presumibilmente con altre risorse. Tale condivisione presuppone una relazione di fiducia tra le parti, misurata in grado di trust.
Il trust qui considerato non è quindi definito a priori ma è costituito gradualmente dai rapporti di fiducia durante le interazioni tra le entità. In questo modulo di interazione infatti i soggetti decidono il grado di fiducia da assegnare alle altre entità in base all'obiettivo. Le informazioni di trust poi verranno conservate e, di volta in volta aggiornate nella fase 4.
La rete schematizzata nel modulo 3, come detto, sarà costituita dai nodi rappresentanti le risorse umane e da archi attraverso i quali si trasmettono i gradi di fiducia. Le risorse, a seconda del ruolo all'interno della fase di valutazione si suddividono in *trustor* (soggetto che pone la fiducia e dà una valutazione) e *trustee* (entità su cui si pone la fiducia e riceve la valutazione). Questi ruoli sono ovviamente intercambiabili, a seconda della fase di valutazione.

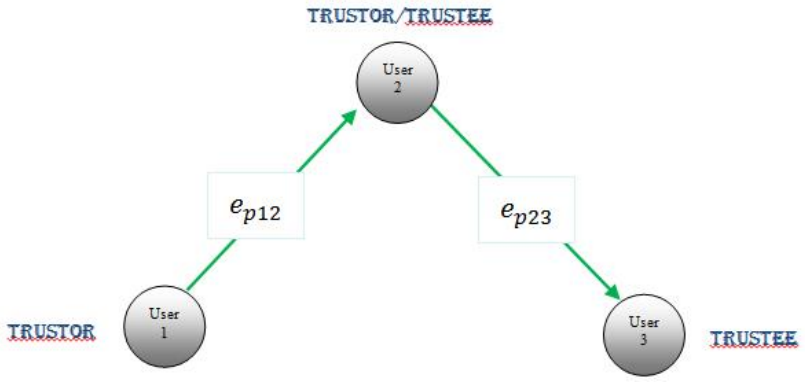

Figura 29. Valutazioni tra Trustor e Trustee

La relazione di fiducia è rappresentata come un arco orientato dal trustor verso il trustee. Ad ogni collaborazione avvenuta tra due risorse corrisponderà un arco tra i nodi che le rappresentano. Il grado di affidabilità è rappresentato da un valore positivo e_p (o negativo e_n), apposto come etichetta

dell'arco. Vedremo poi come questi valori (compresi tra 0 e 1) saranno utili nel calcolare il grado di trust (o untrust) nella fase fondante del modulo.
La rete generale di riferimento, considerata in fase 2, sarà differente da quella appena illustrata. In essa infatti saranno rappresentati altri nodi di tipo competenza, a cui ogni user potrà essere connesso mediante un arco che indicherà il possesso di tale competenza (fig. 30).

5.3.1 Come formalizzare il trust

L'idea circa il concetto di fiducia nel nostro ambiente collaborativo è qui di seguito meglio spiegata.
Assumiamo che il trustor sia *User1* e il trustee sia *User3*. Se c'è stata almeno una diretta esperienza nella quale *User1* abbia collaborato con *User3* allora il grado di trust è calcolato in modo diretto (dall'analisi delle valutazioni a valle delle interazioni tra *User1* e *User3*), altrimenti il grado di trust è computato tramite la sua propagazione sulla rete.
Nel caso della propagazione, per esempio, assumiamo l'esistenza di un terzo lavoratore (*User2*). *User2* trusts *User3* in uno specifico task, *User1* trust *User2* circa la sua opinione nelle specifiche abilità summenzionate, cosicché sarà possibile calcolare il trust di *User1* nella specifica competenza di *User3*.
In un particolare studio (Huang, 2010), è stato costruito un modello basato su formalismi semantici per il calcolo del trust e ci si riferisce a due tipi base di trust: trust in performance e trust in convincimento (rispettivamente nell'aspettativa di performance o nella fiducia).
Il primo tipo di trust consiste in ciò che il fiduciario (trustee) compie, mentre il secondo tipo consta in ciò che il fiduciario crede/pensa. Il mio lavoro prende spunto dai risultati di tale modello e li estende al fine di trattare il concetto di trust nel modello di team formation. In particolare, proponiamo le seguenti definizioni:

$$trust_{per}(u, v, use(v, comp)); \quad (5.1)$$
$$trust_{cred}(v, u, use(z, comp)). \quad (5.2)$$

La funzione *use(u, x)* indica la capacità da parte dell'utente *u* di applicare la competenza *x*.
La definizione (5.1), relativa al ***trust in performance***, indica che l'utente *u* (trustor) ha fiducia nell'utente *v* (trustee) circa la capacità di *v* di utilizzare la competenza *comp*.
La definizione (5.2), relativa al ***trust in convincimento***, indica che l'utente *v* (trustor) ha fiducia nell'utente *u* (trustee) circa le credenze di *u* sulla capacità da parte di *z* di utilizzare la competenza *comp*. Quest'ultima sarà molto utile per il calcolo del grado di trust per propagazione.
Nel prossimo paragrafo vedremo come è possibile calcolare il valore di tali funzioni.

5.3.2 Calcolo del Trust, Distrust e Untrust

Matematicamente il trust è definito come una "probabilità soggettiva" con cui un agente compie una determinata azione, vediamo ora il formalismo che traduce tale affermazione.
L'approccio probabilistico illustrato da Huang et all (2010) è stato adattato al nostro contesto e orientato al nostro obiettivo, costituendo in realtà una evoluzione per lo stesso riferimento.
In questa sezione è proposto un metodo per il calcolo del grado di trust. Inoltre sono aggiunte altre due misure: il distrust e l'untrust.
La fiducia può essere anche interpretata come una distribuzione probabilistica su tre stati mentali (credito, sfiducia e indecisione) corrispondenti a *trust*, *distrust* e *untrust*.

- Credito. Il ***trust*** α di *e* (trustor) per *d* (trustee) nell'applicare con successo una competenza *x* nel contesto *k* può essere approssimato con la probabilità che *d* applicherà con successo *x* nel contesto *k*.
- Sfiducia. Il ***distrust*** β è l'opposto di trust.

- Indecisione. L'***untrust*** γ potrebbe essere calcolato usando la formula α + β + γ = 1, corrispondente ad una distribuzione di probabilità.

Tali valori possono essere calcolati in termini di frequenze di esperienze positive tra trustor e trustee. Più in dettaglio, nel caso del trust in performance può essere usata la seguente formula:

$$\alpha = td_{per}(d, e, use(e, x), k) = n/m. \qquad (5.3)$$

Dove *m* è il numero totale di situazioni valutate da *d* (trustor), nelle quali *e* (trustee) ha applicato la competenza *x* nel contesto *k, ed n* è il numero delle esperienze positive in quel gruppo.
Il distrust (β) può essere calcolato con la seguente formula:

$$\beta = dtd_{per}(d, e, use(e, x), k) = l/m. \qquad (5.4)$$

Dove *l* è il numero delle esperienze negative nello stesso gruppo. Grazie a quanto assunto nel paragrafo 5.2.2, i valori di *n* ed *l* possono essere calcolati con le seguenti espressioni:

$$n = \sum_{i=1}^{m} e_p(\mathrm{i}), \quad l = \sum_{i=1}^{m} e_n(\mathrm{i}) \qquad (5.5)$$

In cui $e_p(i)$, $e_n(\mathrm{i}) \in [0,1]$, rappresentano il grado delle esperienze *i* rispettivamente positivo e negativo, inoltre $e_p(i)+e_n(\mathrm{i}) \leq 1$.
L'esperienza positiva (o negativa) di un trustor rispetto ad un trustee è rappresentata da un task, che richiede competenza *x*, dove il supervisore *d* (trustor) ha fornito un valore di revisione $e_p(i)$ (o $e_n(\mathrm{i})$) positivo (o negativo) al dipendente *e* (trustee) per l'impiego della competenza *x* nel task k.
I valori $e_p(i)$, $e_n(\mathrm{i})$ sono assegnati, mediante le valutazioni come rappresentato in figura 29.
Riguardo il caso del trust nel convincimento può essere usata la seguente equazione:

$$\alpha = td_{cred}(g, d, k) = n'/m'. \qquad (5.6)$$

Dove $td_{cred}(g, d, k)$ approssima $td_{cred}(g, d, use(e, x), k)$, *m'* è il numero totale di esperienze osservate da *g* (trustor), nel quale *d* (trustee) ha applicato ognuna delle competenze nel contesto *k,* e *n'* è il numero di esperienze positive in quell'insieme.
β (distrust) può essere calcolato dall'uso della formula (5.6), sostituendo *n'* con *l'*, dove *l'* è il numero delle esperienze negative nello stesso insieme.

$$n' = \sum_{m'}^{i=1} e'_p(i) \quad l' = \sum_{m'}^{i=1} e'_n(i) \qquad (5.7)$$

Dove $e'_p(i)$, $e'_n(i) \in [0,1]$, rappresentano rispettivamente il grado positivo o negativo di esperienze *i*, inoltre $0 \leq e'_p(i) + e'_n(i) \leq 1$.
Similmente al trust in competenza, un'esperienza positiva (negativa) di un trustor con un trustee è rappresentata da un task dove il supervisore *g* (trustor) ha fornito un giudizio positivo (o negativo) $e'_p(i)$ (o $e'_n(i)$) al lavoratore *e*, circa le credenze di *d* (trustee), sulla capacità di *e*, di applicare ogni competenza nel task *k*.

5.3.3 Propagazione del Trust

Visto che nella fase 3 del modello in figura 28 i ruoli (valutatore e valutato) tra le risorse sono intercambiabili, applicando le formule sopra riportate a tutti i nodi del grafo, possiamo etichettare tutti gli archi esistenti nel grafo con dei valori di trust. Gli archi presenti nel social graph, a quanto detto fin ora, rappresentano solo i giudizi tra due risorse che hanno collaborato direttamente, o al più, che hanno una terza risorsa in comune con cui abbiano collaborato. Vediamo invece ora come è possibile aggiungere altri archi nella rete, ricavando quei giudizi di trust anche tra risorse che non hanno mai collaborato, e giungendo ad una struttura fortemente connessa.
Presentiamo un modello di propagazione del grado di trust in un social network (C.-N. Ziegler, 2005) al fine di calcolare il trust in performance tra il dipendente *e* ed il dipendente *d* anche se *e* non ha mai supervisionato un task eseguito da *d*, nel quale *d* ha usato la competenza richiesta *x* nel contesto *k*. In questo caso è possibile sfruttare le caratteristiche della rete sociale aziendale e il concetto di trust nel convincimento per propagare il trust nelle competenze tramite la contestualizzazione dell'approccio presentato nel modello di Huang (2010).
Calcoliamo il trust (ptd_c) e il distrust ($pdtd_c$) nella performance:

$$ptd_{per} = td_{cred}(e,d,k) \cdot td_{per}(d, g, use(x, k)) + dtd_{cred}(e,d,k) \cdot dtd_{per}(d, g, use(x, k)) \quad (5.8)$$

$$pdtd_{per} = dtd_{cred}(e,d,k) \cdot td_{per}(d, g, use(x, k)) + td_{cred}(e,d,k) \cdot dtd_{per}(d, g, use(x, k)) \quad (5.9)$$

Le formule 5.8 e 5.9 possono essere usate per propagare il trust, il distrust e l'untrust, considerando i percorsi di ogni lunghezza sulla rete sociale aziendale.
Nella figura 30 è illustrata una porzione della rete aziendale in cui è possibile sfruttare, nel contesto k, il concetto di trust nel convincimento per la propagazione, sopperendo alla mancanza di informazioni riguardo il nodo di riferimento.
L'idea di contesto, che nel nostro caso rappresenta il ticket (ossia uno o più task), è ben descritta nel cap 5.3. Per ogni ciclo di fase 3 dello schema in figura 28 ci sarà un grafo che rappresenterà uno o più task, ossia un grafo per ogni ticket.

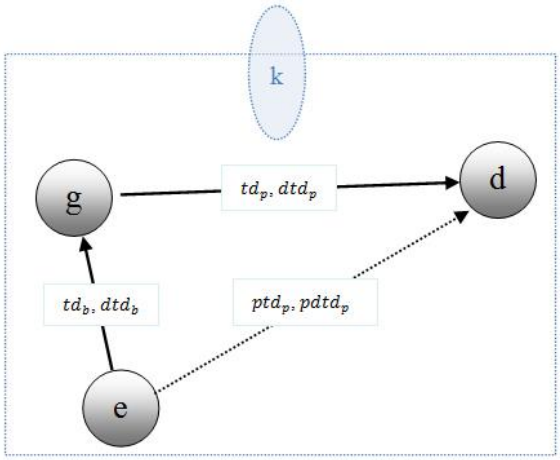

Figura 30 - Propagazione del trust in competenza. Nell'esempio sono indicati solo i gradi per il trust (*td*) e il distrust (*dtd*). In particolare td_p e dtd_p equivalgono a $td_{per}(d,g,use(x,k))$ e $dtd_{per}(d,g,use(x,k))$, mentre td_b e dtd_b equivalgono a $td_{cred}(e,g,k)$ e $dtd_{cred}(e,g,k)$. ptd_p e $pdtd_p$ sono valori calcolati per propagazione, usando le formule 5.8 e 5.9

In particolare, nella figura 30, per il dipendente *e* non ci sono valori di trust, distrust e untrust, relativi alla competenza *x* dal dipendente *d* nel contesto *k*. E' possibile però ottenere il valore summenzionato dallo sfruttamento del trust nella fiducia (nel conteso k) di *e* nei confronti di *g* e il trust nella competenza *x* da *g* per *d* nel contesto *k*. Le competenze che occorrono sono ricavabili dalla conoscenza del contesto, in questo caso *k*.
Il trust in convincimento è propagato dalla sostituzione, in (5.8), di $td_{per}(d, g, apply(x, k))$ e $dtd_{per}(d,g, apply(x, k))$ con $td_{cred}(d,g,k)$ e $dtd_{cred}(d,g,k)$. Allo stesso modo otteniamo la propagazione del grado di distrust, modificando opportunamente la formula (5.9).
Quindi possiamo riassumere affermando che, per propagazione è possibile popolare tutti gli archi della rete con valori di trust. In particolar modo, per il trust in competenza, si sfruttano sia i valori delle performance che quelli del convincimento. Per calcolare la propagazione del trust in convincimento si utilizzano unicamente i singoli valori del trust in convincimento.

5.4 Aggiornamento profili

La fase 4 del flusso disegnato in Fig. 28, a valle della costruzione del grado di fiducia, consta nell'aggiornamento dei profili-utenti e delle tabella relativa ai task.
Riguardo quest'ultima, la fase 4 consiste in un mero refresh dei campi relativi alle competenze richieste per ogni task, in base a quelli già risolti (tab. 11).
Riguardo invece alle tabelle relative ai profili delle risorse, quelle coinvolte nelle precedenti assegnazioni, oltre ad un eventuale aggiornamento delle competenze (tab. 11), saranno oggetto dei feedback da parte dei colleghi, riguardo i gradi di trust. Tale aggiunta può essere sia incrementale, nel caso di acquisizione di una nuova competenza, che regressiva nel caso in cui si riscontrassero collaborazioni negative tra utenti.
Vediamo come la tabella 10 viene aggiornata, ad esempio, nel caso della risorsa User1, skillato in linguaggio java (competenza c_1) e modellazione logica (competenza c_2):

User 1	*User 2*	*User 3*	*User 4*
c_1	α=0.6 β=0.1	α=0.5 β=0.4	α=0.4 β=0.1
c_2	α=0.5 β=0.1	α=0.5 β=0.5	α=0.8 β=0.0

Tabella 12. Gradi di trust (trust e distrust) da parte di User2, User3 e User4 nei confronti di User1 per le competenze c_1 e c_2.

5.5 Team Formation sulle collaborazioni e sulle competenze

Definiti i profili (fase 1) in base alle relazioni di trust (fase 3), partendo dall'approccio seguito nel Cap. 3.4, nella fase 2 ci proponiamo di risolvere il problema del team formation in un contesto aziendale al fine di completare un progetto, in cui sono necessari un numero prefissato di skill richiesti. L'idea è quella di trovare un gruppo di lavoratori che allo stesso tempo soddisfino i requisiti minimi e lavorino "efficacemente" insieme. L'efficienza potrebbe essere misurata in termini di costi di comunicazione e di collaborazione tra i lavoratori, i quali, secondo la nostra teoria si riflettono nel grado di trust (fiducia) che un lavoratore ha nelle competenze di un suo collega.
Fondamentalmente basiamo la costruzione del modulo 2 della nostra metodologia (fig. 28) sui risultati di Huang descritti nel paragrafo 5.2.3, dove il grado del *trust*, *distrust* e *untrust* sono definiti,

rispettivamente, come α, β e γ, con α + β + γ = 1, e le definizioni di Capuano et all (cap. 5.2.4) di trust nella performance e trust nel convincimento, rispettivamente (5.3) e (5.6).
La relazione-peso tra i lavoratori è costruita sul concetto di trust, il grado del quale è espresso come un valore dentro l'intervallo normalizzato [0,1].
Creiamo un grafo direzionato G(W,E), dove W indica il numero dei lavoratori o nodi, ed E le connessioni tra essi. Tale grafo è differente rispetto a quello utilizzato per la creazione delle relazioni di fiducia, illustrato nel cap. 5.3.
Le competenze tra i lavoratori sono mappate nel set delle competenze richieste, riflettendo un livello fissato (dal task), cioè il livello delle competenze di un lavoratore, precedentemente storato, deve essere uguale o maggiore di quelle richieste dal task.
In figura 31 mostriamo un esempio del grafo orientato di riferimento, in cui sono rappresentate le risorse, le competenze e le relazioni tra esse.

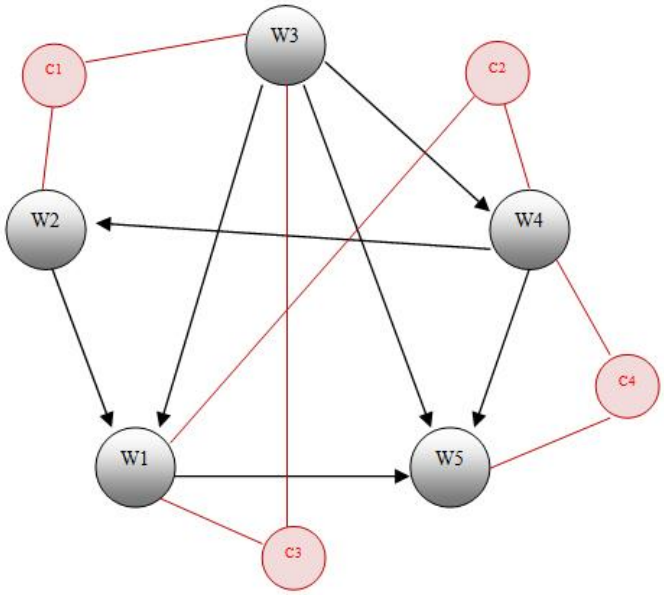

Figura 31. Grafo direzionato di riferimento – I nodi in nero rappresentano i professionisti, i nodi in rosso rappresentano le competenze. Le connessioni in verde rappresentano le relazioni tra due colleghi, le connessioni in rosso indicano le competenze dei lavoratori.

Inoltre, se un lavoratore ha più competenze, è possibile modificare il grafo dividendo il nodo-lavoratore in due corrispondenti nodi, ognuno dei quali connesso con una competenza, come mostrato in figura 32.

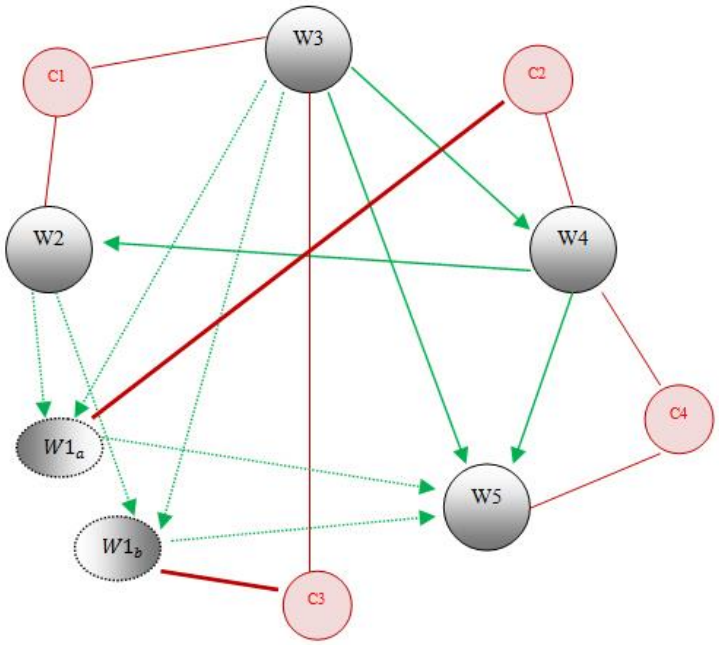

Figura 32. Esempio di nodo W1 con due competenze, scisso in due sotto-nodi corrispondenti. Ogni sotto-nodo sarà connesso alla relativa competenza, mentre gli archi di connessioni sociali verranno duplicati.

5.5.1 Il Modello e l'algoritmo

Per ogni coppia nodi W_i, W_j, ci sarà un arco che li connette con peso t_{ij} funzione della fiducia tra i due lavoratori ossia $t_{ij} = f(\alpha, \beta, \gamma)$.
Tenendo ben presente che α + β + γ = 1, nel caso di nodi disgiunti, allora γ = 1, perché tra le due risorse non c'è alcun trust, ossia massima incertezza e nessuna possibilità di valutazioni pregresse. A tali archi daremo un peso massimo pari a ∞ .
Per tutti i restanti archi, dobbiamo decidere se fare rifermento al valore del trust (α) o al suo inverso (β), a seconda se vogliamo costruire rispettivamente un problema di massimizzazione o di minimizzazione. Per le motivazioni ben spiegate nella parte introduttiva al team formation (cap. 3.1.3) decidiamo di considerare, come riferimento, il valore di β ossia l'untrust. Pertanto, l'obiettivo finale del nostro problema di team formation, dopo questa formalizzazione, diventa di minimizzazione del grado totale di untrust e quindi si traduce nel trovare il team con il minore grado di sfiducia possibile ossia con le migliori relazioni tra i suoi componenti. In questo contesto dobbiamo sempre tener presente che bisogna ricercare il team che abbia le competenze richieste per cui, tenendo sempre presente il modello generico (cap 3.1.4), modifichiamo il modello attuale formalizzato nel cap 4.3 come segue:

L'insieme W=$\{W_1, \dots, W_n\}$, è costituito dalle risorse a disposizione, ognuna con delle competenze C_p=$\{C_{p1}, \dots, C_{pm}\}$; la funzione $x(W_i, C_{pi})$ restituisce il grado di competenza (sotto forma di valore booleano) della risorsa W_i nella competenza C_{pi}. La variabile decisionale t_{ij}, che stabilisce il peso della collaborazione tra le due risorse W_i e W_j, è settata nel seguente modo:

$$t_{ij} = f(\alpha, \beta, \gamma) = \begin{cases} \infty \text{ se } \gamma = 1 \\ \beta \text{ se } \gamma \neq 1 \end{cases}$$

Funzione Obiettivo. Bisogna coprire l'insieme delle competenze richieste C_p ottenendo il minor grado possibile di untrust, ovvero minimizzare il valore β di sfiducia complessiva. Il problema può essere enunciato come segue: "*Dato un set di n individui W ={1, … , n }, un grafo G (W,E), e task T, troviamo* $\bigcup_i skill(W_i, C_{pi})$, *cosicché* $x(W_i, C_{pi}) = 1$ *e il costo di collaborazione* β *sia minimo*". La funzione obiettivo quindi sarà della forma:

$$Assegnamento = min\ (\beta)$$

vincolata da:

$$\forall C_{pi} \in C \quad \sum_{W_i \in W} x(W_i, C_{pi}) \neq \text{null} \qquad \text{(v5.1)}$$

$$\forall i \quad S \subseteq \bigcup_i skill(W_i, C_{pi}) \qquad \text{(v5.2)}$$

$$\forall\ i, j \qquad \beta(W_i, W_j) \leq y \qquad \text{(v5.3)}$$

Il vincolo (v5.1) garantisce che ci sia almeno una risorsa con le competenze richieste, mentre il vincolo (v5.2) stabilisce che l'insieme delle abilità possedute sia almeno uguale all'insieme delle abilità richieste *S*. Il responsabile conosce l'insieme *S* grazie alla tassonomia descritta nella tabella 11, mentre *skill*(W_i, C_{pi}) restituisce le abilità del dipendente W_i estratte per la competenza C_{pi} rispetto alla tabella RDF.
La variabile, *y* introdotta nella (v5.3), a discrezione del supervisore, limita superiormente il numero di cattive collaborazioni pregresse. Praticamente questo vincolo ci consentirà di collegare oppure no due dipendenti in base alla soglia scelta del parametro di trust.
Per l'implementazione relativa al modello sopra descritto, applichiamo una versione modificata dell'algoritmo Directed Steiner Tree (Charikar, 1999), in cui nodi terminali sono le competenze.
L'algoritmo prende in input un set di lavoratori, W=$\{W_1, \dots, W_n\}$, un insieme di competenze richieste, C_p=$\{C_{p1}, \dots, C_{pm}\}$ e il grafo G (W,E), mentre l'output è il sottoinsieme T⊆W. Prendendo a riferimento quanto descritto nel cap 3.1.3, il grafo risulta dello stesso tipo illustrato in figura 8. Ogni competenza-richiesta è mappata sui lavoratori per ottenere un grafo simile a quello in figura 33. Se un lavoratore è connesso a più competenze, viene costruito il grafo in figura 34, associando nello stesso tempo i pesi ad ogni arco. Fissato H come il grafo in cui i terminali sono l'insieme delle competenze e la radice è una competenza, gli step operativi sono descritti dall'Algoritmo 11.

Algoritmo 11. Directed Steiner Tree modificato

1. **Input**: Social Network G(W,E); dipendenti W={W_1, …, W_n}; task T={i_1, … , i_k}; competenze C_p={C_{p1}, …,C_{pm}}; grafo H in cui le foglie sommano le competenze mentre la radice è una competenza.
2. **Output**: Team T⊆W.
3. Select $C_{pi} \in C_p$
4. $H \leftarrow C_{pi}$
5. while C_p is not empty
6. Select $C_{pi} \in C_p$
7. ***Find minimum path (p, C_{pj}, H)*** *// Classico Algoritmo di Djkstra*
8. if (p not exist)
9. H ← C_{pj}
10. else
11. H ← path p
12. Return T⊆W

Il core dell'algoritmo è costituito dalla ricerca del percorso minimo tra le competenze.
Abbiamo deciso di risolvere tale problematica mediante l'elegante algoritmo di Djkstra.
La complessità di questo noto algoritmo è O(n^2) dato che nel caso peggiore il nodo d'interesse viene raggiunto per ultimo, e quindi dobbiamo eseguire n passi, ciascuno dei quali richiede O(n) iterazioni.
Il tutto è ripetuto per il numero di competenze richieste, quindi possiamo affermare che la complessità è O($m \times n^2$).

6. Risultati sperimentali

In questa sezione verifichiamo l'applicabilità del nostro modello risolutivo in contesti reali.
Dapprima illustriamo uno scenario concreto e mostriamo i vari step dell'algoritmo risolutivo.
Successivamente abbiamo introdotto dei risultati sperimentali circa gli esiti dell'esecuzione del nostro modello su grosse quantità di dati reali tratto da un Database Oracle.
Infine illustreremo i vantaggi apportati dal nostro metodo ma anche le migliorie pensabili e gli sviluppi futuri.

6.1 Caso d'uso

Osserviamo come opera il nostro modello nel seguente scenario reale:

Al mio Ufficio afferiscono un insieme di risorse $\{W_1, W_2, \ldots, W_n\}$ che posseggono ognuna delle competenze $\{C_{p1}, C_{p2}, \ldots, C_{pm}\}$.
Dalla piattaforma di Helpdesk giunge un ticket che ci introduce nel nostro contesto k, dando il via alla fase 1 del flusso rappresentato in figura 28.
Il ticket, caratterizzato dal task t_1, viene analizzato.
Secondo la tabella delle attività (simile alla tabella 11), la risoluzione di t_1 richiede il seguente insieme C di competenze $\{C_{p1}, C_{p2}, C_{p3}, C_{p4}\}$.
Il responsabile, utilizzando la tabella dei profili (simile alla tabella 10), filtra tutte le risorse che abbiano almeno una delle succitate competenze e per ognuna di esse, seleziona solo le competenze che occorrono:

W_1
C_{p1}
C_{p2}
C_{p3}

W_4
C_{p1}
C_{p3}
C_{p4}

W_5
C_{p1}

W_7
C_{p2}
C_{p3}
C_{p4}

A questo punto vengono utilizzate le tabelle (tab. 12), viste nella fase della descrizione dei profili, per estrapolare i valori di fiducia. In questo caso decidiamo di minimizzare il valore della sfiducia, quindi estrapoliamo solo i valori dell'untrust (β).

W_1	β
C_{p1}	0.2
C_{p2}	0.3
C_{p3}	0.1

W_4	β
C_{p1}	0.4
C_{p3}	0.1
C_{p4}	0.2

W_5	β
C_{p1}	0.1

W_7	β
C_{p2}	0.1
C_{p3}	0.2
C_{p4}	0.1

Nell'ottica del rispetto del vincolo (v5.3) del modello, il supervisore, sceglie di eliminare tutte quelle relazioni che abbiano valore di sfiducia superiore a 0.2, ottenendo il seguente risultato grafico:

W=$\{W_1, W_4, W_5, W_7\}$;
task T=$\{t_1\}$;
competenze $C_p = \{C_{p1}, C_{p2}, C_{p3}, C_{p4}\}$

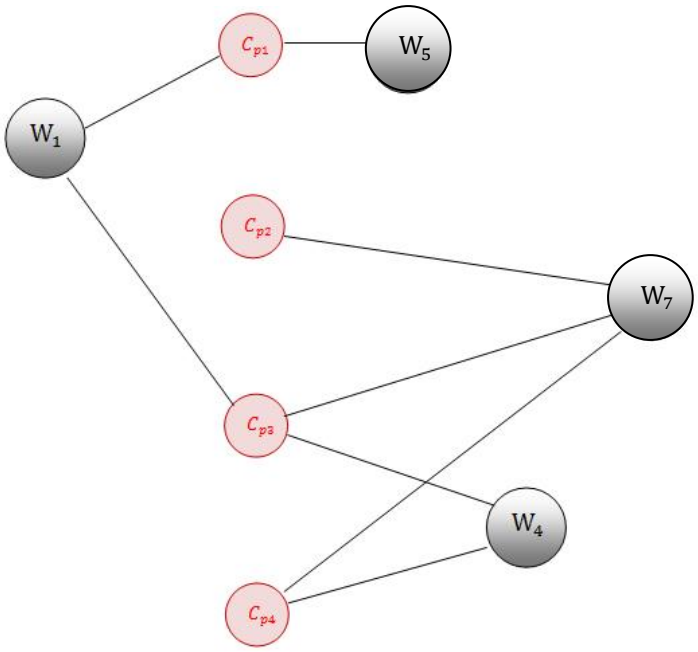

Figura 33. Grafo User-Competenze con $\beta \leq 0.2$

La fase 2 riceve in input il grafo di figura 33.
Applicando l'algoritmo 11 otterrò:

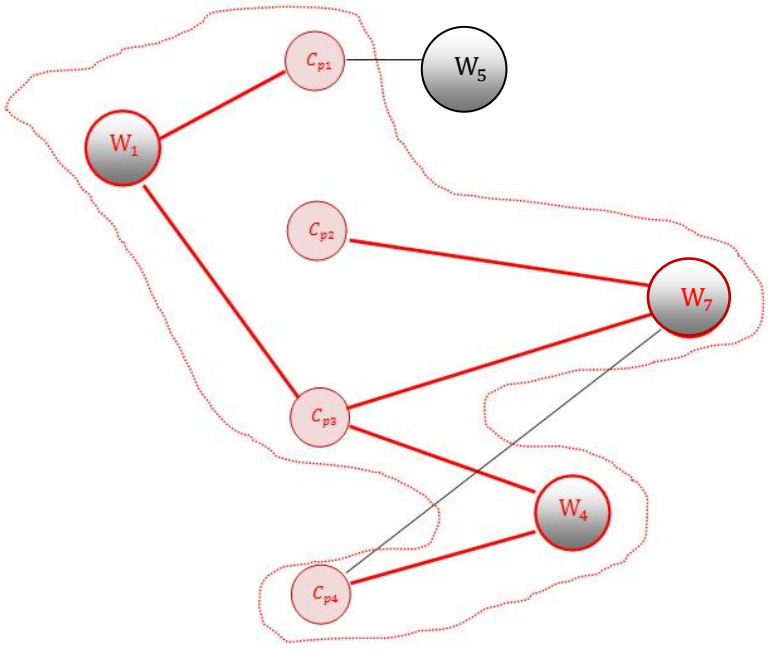

Figura 34. Esecuzione dell'algoritmo 11

Per cui il team risultante, in grado di risolvere il task t_1, minimizzando il tasso di insoddisfazione, sarà $\boldsymbol{T_f = \{W_1, W_4, W_7\};}$
Alla fase 3 del flusso di figura 28 sarà restituito il seguente input sul task t_1:

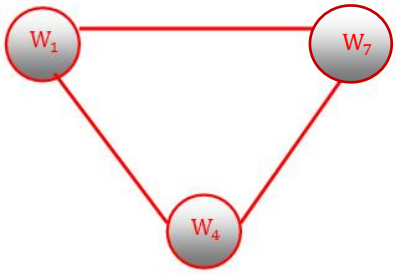

Figura 35. Grafo input alla fase 3

In questa fase di creazione delle relazioni di fiducia, saranno etichettati gli archi con i valori di trust secondo le formule di cap 5.3.
La dove c'è stata una collaborazione diretta saranno utilizzate le (5.5).
Dove invece non vi è stata alcuna competenza in comune, e quindi nessuna collaborazione diretta, verrà utilizzato il concetto di propagazione sfruttando la nozione di trust nel convincimento (5.8) e (5.9).
Così facendo, utilizzando il concetto di feedback, si otterranno valutazioni di β del tipo:

$W_1 \rightarrow 0.2 \rightarrow W_4$
$W_1 \rightarrow 0.1 \rightarrow W_7$
$W_4 \rightarrow 0.5 \rightarrow W_1$
$W_4 \rightarrow 0.3 \rightarrow W_7$
$W_7 \rightarrow 0.2 \rightarrow W_1$
$W_7 \rightarrow 0.1 \rightarrow W_4$

Tali valori verranno dati in input alla fase 4 in cui saranno utilizzati per aggiornare le tabelle illustrate in cap 5.3

6.2 Sperimentazione

In questa breve sezione valutiamo e compariamo la performance del modello proposto per trovare team utili a risolvere determinati task.
Mostriamo che il nuovo metodo può portare qualità ed efficienza non solo in termini di costo di comunicazione, cardinalità del team, ma anche dal punto di vista del tempo di esecuzione.
In realtà per testare il modello su grandi quantità di dati è stata utilizzata, in prestito, una base dati relazionale "carrieristica", estratta da un Database principale Oracle residente negli uffici presso cui presto servizio. Tale Database è stato modificato all'occorrenza, facendone un'istantanea ad una certa data di estrazione e soprattutto rendendo irriconoscibili, ai fini della salvaguardia della privacy, i dati anagrafici e quelli sensibili in generale.
Il Db contiene dati relativi alle carriere di alcuni particolari professionisti (Segretari Comunali e Provinciali) nonché le informazioni relative alle segreterie (Comuni o Provincie). Si è supposto che tali professionisti fossero le risorse umane del social come visto nei capitoli precedenti. Le relazioni tra le risorse sono individuate attraverso le collaborazioni che esse hanno avuto in passato con le varie segreterie degli enti locali.
La social network basata sul concetto di trust è stata realizzata come segue.
Visto che il Db è stato progettato su due entità fondamentali "Segretari" e "Segreterie", considereremo, nella rete, collegati direttamente, due Segretari che in passato hanno lavorato ad una

stessa segreteria. Per effettuare il calcolo del grado di trust, contestualizzandolo nell'ambito lavorativo sopra descritto, abbiamo inoltre deciso di tarare il grado di trust in base alla lunghezza del periodo di nomina. Infatti se un Segretario Comunale è impiegato presso un Comune per molti anni, sopravvivendo addirittura a più legislature, a più Sindaci e a Giunte diverse, si sottende che egli abbia lavorato bene ed abbia goduto del massimo grado di fiducia da parte dei suoi collaboratori. Verissimo anche l'inverso!
In tal modo baipassiamo tutta la fase della creazione delle relazioni di fiducia (osservata attentamente nello studio del caso d'uso par. 6.1) e ci concentriamo sul rendimento del nostro modello per la formazione di team validi.
I professionisti sono definiti skillati se nei loro profili (curriculum) ci sono determinati requisiti richiesti dalle norme vigenti per poter esercitare lo specifico ruolo in particolari classi di segreterie.
In fine, per validare il metodo di team formation, immaginiamo che più Segretari possano collaborare ad un'unica segreteria (anche se in realtà è vero l'inverso!), fino a saturare il numero di skill richiesti per quella segreteria.
In totale, nel dataset considerato, alla data di riferimento, ci sono 3589 segretari, 7807 segreterie, e 32526 connessioni.
I pesi sugli archi (il trust) sono pari al numero di mesi lavorati in ogni segreteria.
Mostriamo il costo di comunicazione, la cardinalità del team, e il tempo di esecuzione. Ogni task generato (richiesta di segretari in una determinata sede) è stato "scisso" nella richiesta di alcuni parametri essenzialmente raggruppati in due categorie: anni di esperienza (dalla data di iscrizione all'Albo); superamento specifici corsi di aggiornamento.
Per ogni insieme di requisiti richiesti sono stati generati i relativi task. Riportiamo i risultati medi ottenuti dai 2 metodi descritti rispettivamente nel cap.4 e nel cap.5.

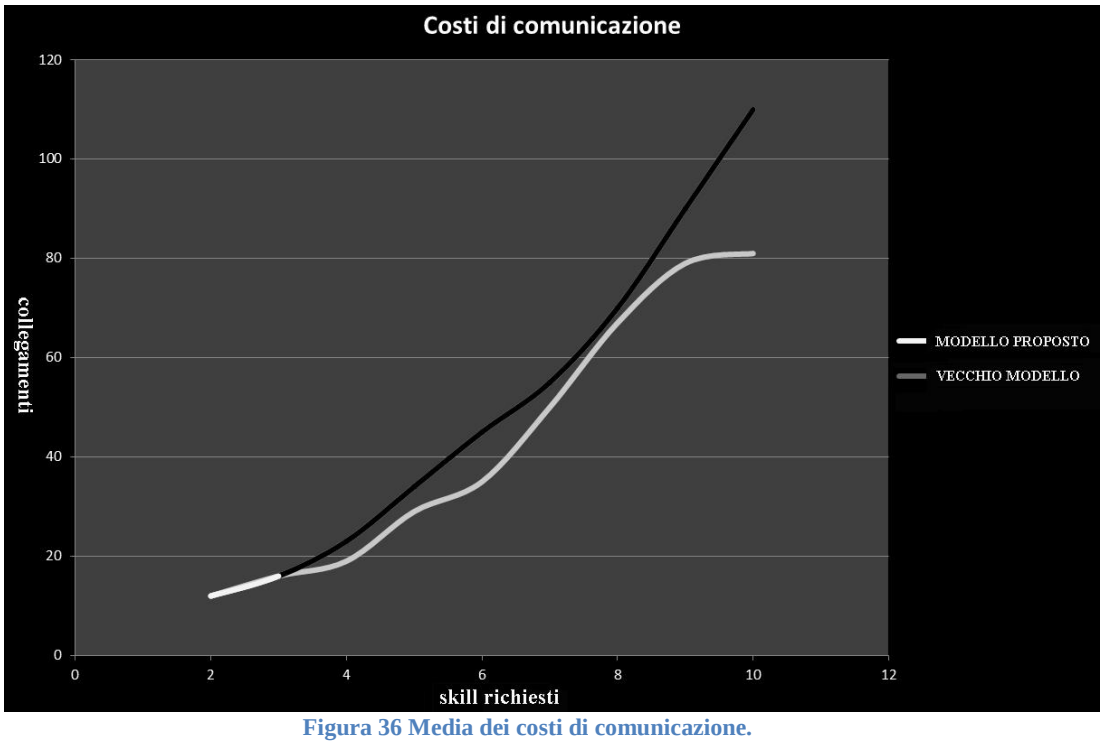

Figura 36 Media dei costi di comunicazione.

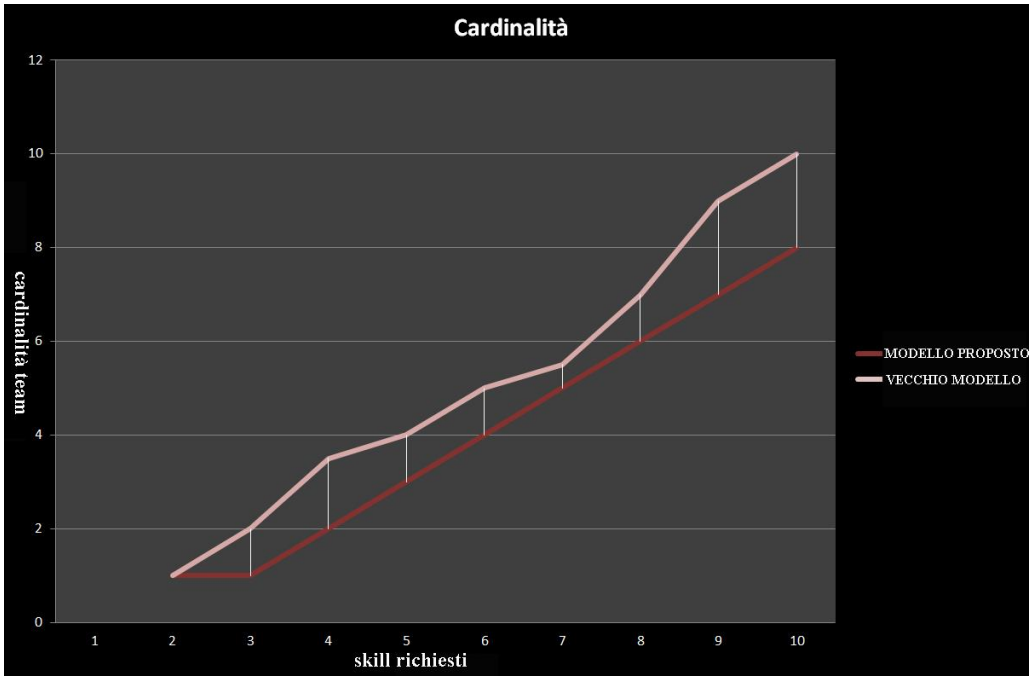

Figura 37 Media della cardinalità dei team.

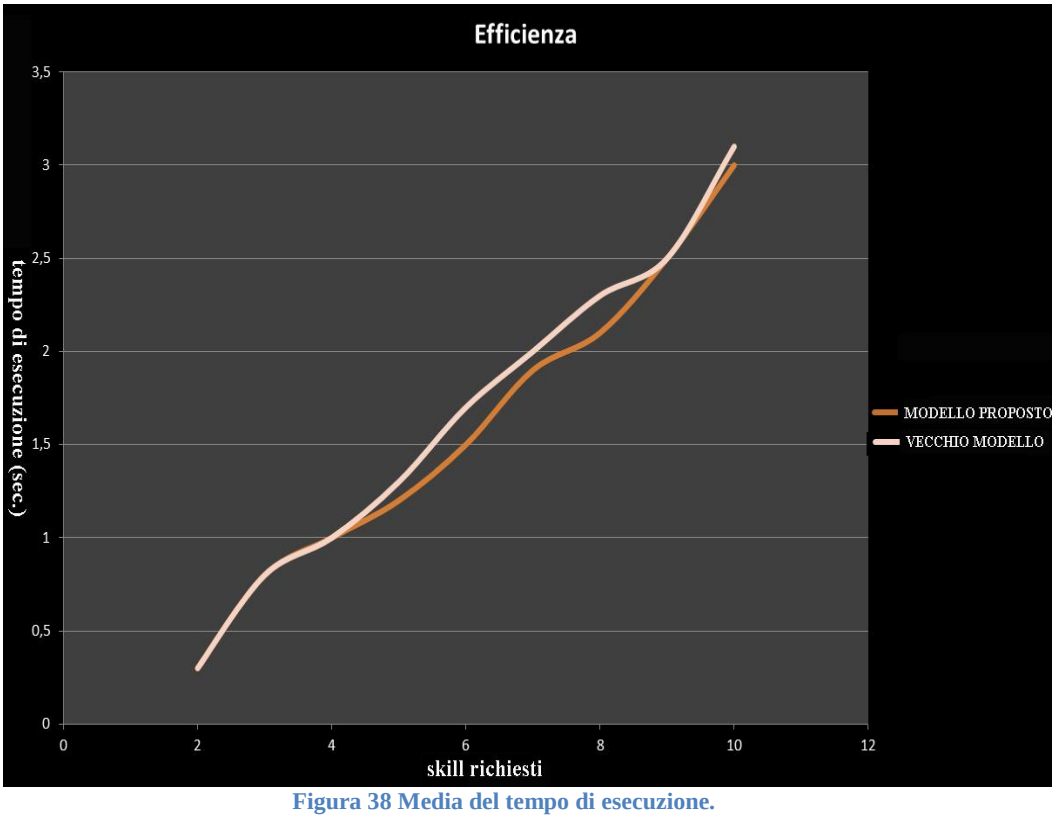

Figura 38 Media del tempo di esecuzione.

Le Figure 36 e 37 mostrano che il metodo proposto supera il vecchio algoritmo utilizzato fino ad oggi nel nostro particolare contesto, sia per quanto concerne il costo di comunicazione che per la cardinalità del team.
In special modo, riguardo i costi di comunicazione, se il numero di skills richiesti cresce, l'efficienza del nuovo metodo è ancor più ragguardevole.
Inoltre, relativamente al modello proposto, possiamo osservare come il numero medio dei membri dei team non aumenta in maniera eccessiva all'aumentare del numero degli skills richiesti.
Riguardo i tempi di esecuzione, i due modelli si comportano in larga parte allo stesso modo, anche se, per alcuni tratti il modello proposto, offre, seppur minimi, vantaggi.
Anche se il vero concetto innovativo è stato quello di introdurre il "grado di trust" per valutare e considerare le collaborazioni, possiamo affermare che altresì dal punto di vista dell'efficienza il nostro modello risolutivo offre notevoli spunti di interesse.

6.3 Vantaggi dell'approccio proposto e migliorie

L'approccio proposto introduce finalmente concetti motivazionali nel processo di selezione di un team di lavoro. L'uso delle funzioni di trust a mio avviso permette un approccio più realistico alla formazione di team di lavoro, consentendo al decision-maker di massimizzare le potenzialità offerte dal proprio team di lavoro.
Il modello proposto è da considerare come una base di partenza su cui poi applicare diverse modifiche in base alle esigenze.
Ad esempio, nella fase di costruzione delle tabelle relative ai profili utenti (tab. 10), potrebbero essere introdotti dei valori numerici che descriverebbero la percentuale di afferenza di una particolare abilità ad un determinato skill.
Nella classificazione dei task (tab. 11), potrebbe essere aggiunta una scala graduata che descriverebbe il grado di difficoltà di risoluzione di una determinata attività. In questo modo nella rete generale di riferimento, considerata in fase 2, gli archi di connessione indicheranno anche il grado di possesso di tale competenza.

6.4 Sviluppi futuri

Lo studio proposto resta sempre aperto a numerosi sviluppi ed ampliamenti.
Tra questi, quelli che potrebbero risultare di maggior rilievo sono di seguito accennati.
Nello schema relativo al flusso logico potrebbe essere ampliata la fase relativa al gap competence nel caso di reclutamento o formazione relativa al raggiungimento di vari gap.
Potrebbe essere utile introdurre dei vincoli sugli skill e creare dei campi relativi ad essi da inserire nella tabella dei profili.
Altri vincoli di upper bound potrebbero essere inseriti sulla taglia del team con la considerazione di un eventuale studio con approccio multi obiettivo.
Una onesta generalizzazione del problema potrebbe essere la sua variante graduata. In una simile variante, il grado delle abilità di un individuo e l'estensione allo skill richiesto per il completamento

di un task potrebbero essere modellate sul valore di un peso intero, per es. {0,1, …, λ}. In questo caso, le specifiche del task, per ogni skill richiesto $a_j \in T$, indicano il livello minimo richiesto λ_j.
Un'altra gradazione può essere aggiunta al livello di completamento del task, come segue: un task T è dato dall'insieme delle triple $\{(a_j, k_j, l_j)\}_{j=1}^{p}$, dove $a_j \in A$, specificando che, per finire il task, è richiesto almeno k_j e al più l_j skill di a_j.

Bibliografia.

Aczel J. and Saaty T.L. Procedures for synthesizing ratio judgments [Journal] // Journal of Mathematical Psychology,. - 1983.

Anagnostopoulos A. , Becchetti, L., Castillo, C., Gionis, A., Leonardi, S. Online Team Formation in Social Networks [Journal]. - MSR Cambridge : [s.n.], 2012.

Anagnostopoulos Aris Power in unity: Forming teams in large-scale community systems. [Journal] // CIKM. - 2010.

Ardichvili A., Page, V., Wentling, T. Motivation and barriers to participation in virtual knowledge Management [Journal]. - 2003. - 1 : Vol. 7.

Barabasi R. , Albert, A. Emergence of scaling in random networks [Journal] // Science. - 1999.

Barabasi R. , Albert, A. Statistical mechanics of complex networks. [Journal] // Review of Modern Physics. - 2002.

Breshnan F., Brynjolfsson, E., Hitt, L. M. Information Technology, Workplace Reorganization, and the Demand for Skilled Labor: Firm-Level Evidence. [Journal] // The Quarterly Journal of Economics. - 2002.

Briggs Myers http://www.myersbriggs.org/my-mbti-personality-type/mbti-basics/ [Online] // Myers Briggs type indicator.

C.-N. Ziegler C.N., Lausen, G. Propagation models for trust and distrust in social networks [Journal]. - [s.l.] : Information Systems Frontiers, 2005.

Campanella J. , Vale, J. The Resilient City: How Modern Cities Recover From Disaster [Journal] // Oxford University Press. - [s.l.] : Oxford Univ Pr., 2005.

Capuano N., Gaeta M., Mangione G.R., Orciuoli R., Ritrovato P. Integrating Trust and Competency Management to Improve Learning [Journal] // 2011 11th IEEE International Conference on Advanced Learning Technologies. - 2011.

Carley K. M. and Gasser L. Computational and Organization Theory [Journal] // Multiagent Systems - Modern Approach to Distributed Artificial Intelligence. - 1999.

Cesa-Bianchi N. Il grafo casuale di Erdős-Rényi [Article]. - 2009.

Charikar M., Chekuri, C.,Cheung, T.-Y., Dai, Z., Goel, A., Guha, S., Li, M Approximation Algorithms for Directed Steiner Problems [Journal]. - [s.l.] : Journal of Algorithms, 1999. - 1, pp.73-91 : Vol. 33.

Chhabra M. Das, S., and Szymanski, B. Team Formation in Social Networks [Journal]. - Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY : [s.n.], 2012.

Christianson B., Harbison, W.S. Why isn't trust transitive? [Journal]. - [s.l.] : Computer Science, 1997. - Vol. 1189.

Coase R. H. The Nature of the Firm. [Journal] // Economica. - 1937.

Colombo M. G. & Grilli L. Founders' human capital and the growth of new technology-based firms: A competence-based view [Journal]. - 2005.

Coppola N,,Hiltz, S., Rotter, N. Building trust in virtual teams [Journal]. - [s.l.] : System Sciences, 2000.

Cristini A. , Gaj A., Leoni R. Direct and Indirect Complementarity between Workplace Reorganization and New Tecnology [Journal] // Rivista di Politica Economica. - 2008. - Vol. 98.

Das Sarma A. A sketch-based distance oracle for web-scale graphs [Journal] // WSDM. - 2010.

Datta S., Majumder, A., Naidu, K. Capacitated Team Formation Problem on Social Networks [Journal]. - International conference on Knowledge discovery and data mining : [s.n.], 2012. - Vols. 1005-1013.

Delgado J. Emergence of social conventions in complex networks [Journal] // Artificial Intelligence. - 2002.

Di Pillo G., Grippa L. An Exact Penalty function method with global convergence properties for nonlinear programming problems [Journal]. - [s.l.] : North-Holland, 1986. - 36 : Vol. Mathematical Programming.
Dyer R.F. and Forman, E.H. Group decision support with the analytical hierarchy process. [Journal] // Decision Support Systems. - 1992.
Erdős P. & Rényi, A. On Random Graphs I [Journal]. - [s.l.] : Publicationes Mathematicae, 1959.
Foray Dominique The Economics of Knowledge [Journal]. - 2006.
Frasca P. and Morone A. Innovazione, network di imprese e conoscenza: quale ruolo per la geographical proximity? [Journal] // Quaderni DSEMS. - 2007.
Freeman R. B. & Kleiner, M. & Ostroff, C. The Anatomy of Employee Involvement and Its Effects on Firms and Workers [Article] // NBER Working Paper. - 2000. - 8050.
Glance N. , Hogg T., and Huberman B.A. Computazional ecosystems in a changing environment [Journal] // International Journal of Modern Physics. - 1991.
Golden B.L., Wasil, E.A. and Harker, The Analytical Hierarchy Process: Applications and Studies [Journal]. - 1989.
Grafi teoria dei [Online] // Teoria dei grafi. - http://it.wikipedia.org/wiki/Cricca_(teoria_dei_grafi).
Gravili G., Turati, C. Organizzazione emergente e tecnologie elettroniche di coordinamento [Journal] // La comunicazione nell'economia d'azienda. Processi, strumenti, tecnologie. - 2000.
Hogg T., and Huberman, B.A. Controlling chaos in distribuited systems [Journal] // Men and Cybernetics. - 1991.
Howells Jeremy RL Tacit knowledge, innovation and economic geography [Journal] // Urban Studies. - 2002.
Huang J., Nicol, D. A formal-seantics-based calculus of trust [Journal]. - 2010.
Hunthausen J.M., Truxillo, D.M., Bauer, T.N., Hammer, L.B. A field study of frame-of-reference effects on personality test validity [Journal] // Appl Psycol. - 2003.
Huysman M. H. and De Wit D. Knowledge Sharing in Practice [Journal] // Springer Science & Business Media. - 2002.
Islei G., Lockett, G., Cox, B. and Stratford, M. A decision support system using judgmental modeling: A case of R&D in the pharmaceutical industry. [Journal] // IEEE Transactions on Engineering Management. - 1991.
Khuller S., Raghavachari B. Improved approximation algorithms for uniform connectivity problems. [Article] // Journal of Algorithms. - 1996. - Vol. 21.
Klett Fanny The Design of a Sustainable Competency-Based Human Resources Management : A Holistic Approach [Journal]. - Ilmenau, Germany : Knowledge Management & E-Learning: An International Journal, 2009. - 3 : Vol. 2.
Kolbe http://www.kolbe.com/ [Online] // Colbe Conative Index.
Kusiak A., Zakarian, A. Forming teams: an analytical approach [Journal] // IIE Transactions. - 1999.
Lappas T., Kun Liu, Terzi, E. Finding a Team of Experts in Social Networks [Journal]. - UC Riverside and San Jose, CA, USA : [s.n.], 2009.
Lee Guang-Siang An extension of Stein-Lovász theorem and some of its applications [Article] // Journal of Combinatorial Optimization. - 2013. - 01. - 1 : Vol. 25.
Lengnick-Hall M. L., Lengnick-Hall, C.A., Andrade L.S., Drake B. Strategic human resource management: The evolution of the field [Journal] // Human Resource Management Review. - 2009.
Leoni R. Cristini A. & Gaj A. & Labory S. Flat Hierarchical Structure, Bundles of New Work Practices and Firm Performance [Journal] // Rivista italiana degli economisti. - 2003.
Leoni R. Gaj A. Informal learning and development of competencies: the role of organizational designs. Implication for industrial policies [Journal]. - 2010.
Lovász L. , Plummer, M. D Matching Theory [Book]. - [s.l.] : North-Holland, 1986.

Matthew E. Gaston Marie desJardins - University of Maryland Baltimore County Team Formation in Complex Networks [Journal]. - 2005.
Mazzanti La valutazione economica dei benefici sociali del patrimonio culturale [Journal]. - 2004.
Milgrom P. Roberts, J. The economics of modern manufacturing: technology, strategy, and organization [Journal]. - [s.l.] : American Economic Review, 1990.
Miller H.J. Evolving information processing organizations [Journal] // In Dynamics of Organizations: Computational Modeling and Organizational Theories. - 2001.
Osterman Paul Wage effects of high performance work organization in manufacturing [Journal]. - 2006.
Papadimitriou C.H., Yannakakis, M. On the approximability of trade-offs and optimal access of web sources [Journal]. - [s.l.] : FOX, 2000.
Pisano G. Knowledge, integration, and the locus of learning: An empirical analysis [Journal] // Strategic Management Journal. - 1998.
Rangapuram S. , Bühler, T., Hein, M. Towards Realistic Team Formation in Social Networks based on Densest Subgraphs [Journal]. - 2013.
Saaty T.L. Axiomatic foundation of the analytical hierarchy process. [Journal] // Management Science. - 1986.
Saaty T.L. Decision Making for Leaders, [Journal] // Lifetime Learning. - 1992.
Saaty The Analytical Hierarchy Process [Journal]. - 1981.
Setzer M. Hein and S. Beyond spectral clustering tight relaxations of balanced graph cuts. [Article] // NIPS. - 2011.
Strogatz D. Watts and S. Collective dynamics of small-world networks [Journal] // Nature. - 1998. - 6684 : Vol. 393. - pp. 440–442.
Taylor [Online] // wikipedia. - http://en.wikipedia.org/wiki/Scientific_management.
Teece Chesbrough e Harward Business Review [Journal]. - 1996.
Walsh J.P. Organizational Memory [Journal]. - [s.l.] : Academy of Management. The Academy of Management Review, 1991.
Wikipedia [Online] // Wikipedia. - http://en.wikipedia.org/wiki/Erd%C5%91s%E2%80%93R%C3%A9nyi_model.
Wikipedia [Online] // Wikipedia. - http://it.wikipedia.org/wiki/Algoritmo_ungherese.
Zwick Thomas Technology Use, Organisational Flexibility and Innovation: Evidence for Germany [Journal] // ZEW Discussion Papers. - 2004.