Perfume

Francesco Izzo & Davide Paragona

September 2025

1 Introduzione

Questa relazione descrive un approccio volto a limitare la comparsa di Community Smells attraverso uno strumento che automatizza il processo di risoluzione. Dopo aver introdotto il dominio del problema e le principali soluzioni già presenti in letteratura e in strumenti esistenti, viene presentato Perfume, un prototipo sviluppato con l'intento di fornire una prima risoluzione automatica dei problemi, evitando interventi manuali.

Vengono quindi illustrate le assunzioni adottate per semplificare la fase iniziale di progettazione, le specifiche di funzionamento dello strumento, i risultati ottenuti e i possibili sviluppi futuri.

2 Dominio del problema

Il "Social debt", un concetto emerso nell'ingegneria del software, descrive l'accumulo di costi imprevisti di progetto che derivano da una comunità di sviluppo non ottimale [Tam+13].

Il debito sociale rappresenta l'impatto di decisioni socio tecniche sub-ottimali sull'ambiente di lavoro, sulle persone, sui prodotti software e sulla società [CCS22].

Il Technical Debt, invece, si riferisce a costi futuri derivanti da soluzioni tecnologiche o di codice sub-ottimali [CPS20].

I "community smells" sono una delle principali fonti del debito sociale e sono definiti come "anti-pattern" sociotecnici [CCS22] e "modelli non ottimali" nella struttura organizzativa e sociale di una comunità di sviluppo software [TPK17]. I community smells sono precursori di eventi socio-tecnici negativi che si manifestano in costi aggiuntivi del progetto, ovvero debito sociale [TPK17]. Studi precedenti hanno dimostrato che i community smells possono persino portare all'introduzione di debito tecnico nel codice sorgente [CPS20].

I quattro community smells più studiati e frequenti, presi in considerazione anche dal framework di Perfume, si manifestano sia in contesti aziendali (eccetto il Black Cloud) che in comunità open-source.[CPS20]. Questi sono:

- Organizational Silo: Riguarda l'eccessiva compartimentalizzazione all'interno di un'organizzazione[TPK17].
- Black Cloud: Indica un'atmosfera negativa, con membri che rallentano il lavoro[TPK17].
- Lone Wolf: Descrive un membro che opera in modo isolato, senza collaborare[TPK17].
- Radio Silence: Si riferisce a una mancanza di comunicazione o a ritardi nelle risposte, che possono bloccare le attività della comunità [TPK17].

Si è deciso di optare per questi precisi quattro CS per le stesse ragioni per cui anche altri prototipi di tool hanno scelto di concentrarsi su questi:

- **Semplificazione del problema**: Il problema che intendiamo affrontare, includendo anche la dimensione sociale, rischia di trasformarsi in un progetto eccessivamente lungo da realizzare e, soprattutto, potrebbe generare ulteriori criticità da dover considerare.
- **La frequenza**: Come in altri articoli la possibilità di prendere in esame solo quelli che si trovano più frequentemente porta ad avere una semplificazione in tutte le fasi del progetto.

2.1 Cause

Ci sono diversi tipi di cause che possono portare alla formazione di CS. Tra le varie cause ci sono:

- Il **contesto** è stato il tipo di causa più frequente. Alcune delle cause risiedevano in organizzazioni che utilizzavano processi molto formali e implementavano procedure o standard rigidi. In queste condizioni, i membri del team seguivano le politiche per raggiungere gli obiettivi del gruppo. Questa rigidità formale ha anche plasmato metodi di pensiero e di lavoro inflessibili, scoraggiato l'innovazione e aumentato il tempo di revisione dei compiti. Altre cause contestuali erano la distribuzione geografica dei membri del team e l'assenza di condizioni per favorire la condivisione delle conoscenze. D'altra parte, le organizzazioni informali mostravano caratteristiche contestuali diverse. Ad esempio, i professionisti si trovavano di fronte a una mancanza di protocolli, a strutture organizzative disallineate e a procedure incompatibili.[CCS22]

- La **comunicazione** è stata il secondo tipo di causa più frequente. Queste cause erano legate a una comunicazione scarsa o addirittura completamente assente. Abbiamo anche riscontrato punti deboli e lacune nelle strutture di comunicazione che influenzavano il flusso di informazioni. Alcuni esempi di questi punti deboli includono fonti di informazione non tracciabili e la mancata condivisione delle decisioni relative all'architettura. In termini di lacune, le organizzazioni non fornivano protocolli, né stabilivano spazio e tempo per la condivisione di conoscenze o esperienze, né avevano persone dedicate a distribuire informazioni ad altri team. [CCS22]

- La **composizione** è stata il terzo tipo di causa più frequente. Abbiamo identificato cause legate a problemi nella composizione del team. Le cause indicavano che i diversi livelli di competenza scatenavano conflitti nei processi decisionali e nella condivisione efficace delle conoscenze. Anche il background professionale rappresentava un problema quando le persone che utilizzavano metodi obsoleti non erano in grado di fornire soluzioni innovative. Inoltre, i cambiamenti nella composizione del team facevano sì che i membri più anziani cambiassero il loro comportamento e non offrissero aiuto ai nuovi arrivati.[CCS22]

- Il **coordinamento** è stato il successivo tipo di causa più comune. Queste cause si concentrano sulla congruenza socio-tecnica nei processi di sviluppo del software. Ad esempio, i team di sviluppo del software affrontavano problemi di frammentazione elevata dei compiti, riluttanza a controllare i task e perdita di tempo nell'implementare le modifiche. Altri problemi di coordinamento includevano l'aumento dell'isolamento tra i membri del team e lo scarso coinvolgimento degli sviluppatori.[CCS22]

- Ci sono stati anche diversi tipi di cause meno comuni. La **cooperazione** descriveva situazioni in cui le persone evitavano il supporto o i consigli dei loro compagni di squadra. Includeva anche una mancanza di interesse nell'identificare possibili requisiti mancanti che potevano influenzare i risultati di altri compiti dipendenti. Le cause a **natura duale** includevano situazioni in cui l'ambiente di lavoro o la composizione del team includevano persone che mostravano o praticavano diverse culture organizzative. Ad esempio, il "DevOps clash smell" si verifica quando team distribuiti geograficamente sono in disaccordo a causa dell'eterogeneità di valori e standard.[CCS22]

2.2 Effetti

Analizzando queste cause ci si rende conto come la parte sociale del problema abbia un ruolo chiave nel generare i vari CS. Una volta che i CS si formano, se non vengono risolti in tempo, possono crescere e comportare degli effetti negativi come:

- **Cooperazione**: tipo di effetto più frequente. Problemi legati a mancanza di fiducia, partnership, efficacia collettiva, stagnazione del team e condivisione di informazioni incoerenti.

- **Debito tecnico**: secondo effetto più comune. Colpisce la qualità del software (artefatti difettosi, fallimenti, codice duplicato). Evidenziata la relazione tra debito sociale e debito tecnico.

- **Comunicazione**: terzo effetto più frequente. Problemi dovuti a strutture/processi di condivisione inefficaci, esclusione di sviluppatori da decisioni architetturali, mancanza di canali definiti, ritardi e disinformazione.

- **Contesto**: cambiamenti nei requisiti, aumento di stress, riduzione del senso di responsabilità, frustrazione, decisioni architetturali prese da persone non competenti, assunzioni errate sui tempi.

- **Coordinamento**: difficoltà dei leader nel garantire congruenza socio-tecnica. Conseguenze: rallentamenti, ritardi, problemi manageriali, attività casuali e tempo sprecato.

- **Cognizione**: scarsa attenzione ai bisogni del prodotto e alle decisioni architetturali. Conseguenze: perdita di conoscenza critica, cattiva gestione del know-how, incomprensioni con stakeholder.

- **Effetti economici**: rischi finanziari per l'organizzazione a causa di fallimenti software e clienti insoddisfatti.

- **Conflitti**: differenze professionali che generano scontri interpersonali, frustrazione, demotivazione e disaccordi sui metodi di lavoro.
- **Effetti a duplice natura**: combinazione di problemi tecnici, conflitti e scarso coordinamento. Esempi: code smells non risolti, mancanza di consenso, pochi responsabili sul codice complesso, perdita progressiva di conoscenza aggiornata.

2.3 Struttura organizzativa

Aspetto cruciale che le ricerche hanno sottolineato è come il tipo di struttura organizzativa posso influenzare quali tipologie di CS possono svilupparsi all'interno della stessa.

Comunità → Smell	**Val.1**	**Val.2**	**Val.3**	**p-value**
Formal Group → Bottleneck	0.77	0.91	1.54	0.033
Formal Group → Lone Wolf	0.73	0.88	1.26	0.013
Informal Community → Organisational Silo Effect	0.74	0.94	1.69	0.011
Informal Community → Lone Wolf	0.68	0.82	1.63	0.022
Informal Network → Organisational Silo Effect	0.71	0.85	1.46	0.024
Informal Network → Black Cloud	0.69	0.83	1.55	0.043
Network of Practice → Bottleneck	0.76	0.89	1.59	0.032

Table 1: Relazioni tra tipi di comunità e smell con valori associati, [De +20]

3 Idea

Nel corso delle nostre ricerche sono stati individuati diversi strumenti con obiettivi analoghi, i quali, verosimilmente per motivi legati alla tutela della privacy, hanno adottato l'approccio di operare su dati più facilmente accessibili e disponibili senza implicazioni dirette sulla riservatezza di dipendenti o collaboratori. Tale scelta, tuttavia, comporta l'esclusione di una componente rilevante del problema che si intende affrontare.

Modellazione sociale In questa prospettiva, una delle principali innovazioni introdotte dal presente strumento consiste nell'inclusione della dimensione sociale all'interno del modello: il problema è infatti rappresentato mediante tre grafi, di cui uno dedicato a riflettere lo stato delle relazioni sociali nella struttura organizzativa.

Risoluzione per ottimizzazione La seconda innovazione riguarda l'impiego, per la risoluzione del grafo, di tecniche di ottimizzazione e, in aggiunta, l'integrazione con modelli derivati dall'analisi delle reti sociali. Una volta formalizzato il problema attraverso variabili e vincoli, tali strumenti sono in grado di individuare la soluzione ottimale o, in alternativa, una soluzione prossima all'ottimo.

È opportuno precisare che lo strumento non si propone di rilevare direttamente i **Community Smells**, bensì di intervenire su situazioni in cui tali fenomeni potrebbero emergere. A tal fine, esso si basa sulle azioni di correzione riportate nella Tabella 7 di [CPS20], applicandole in un contesto vincolato che tenga in considerazione non solo le dimensioni lavorative e comunicative, ma anche quelle sociali.

4 Soluzioni prese in considerazione

Nel manuale *Refactoring Community Smells in the Wild: The Practitioner's Field Manual* si osserva come una serie di soluzioni siano state applicate con successo ad alcuni community smell, sulla base del feedback dei practitioner che le hanno sperimentate. I community smell presi in considerazione sono quelli statisticamente più rilevanti, ovvero:

- **Organizational Silos**
- **Black Cloud**
- **Lone Wolf**
- **Radio Silence/Bottleneck**

Per ciascuno di essi sono state adottate determinate soluzioni, come mostrato nella Tabella 2.

Community Smell	**Soluzioni adottate**
Organizational Silos	• Ristrutturazione della community • Creazione di un piano di comunicazione • Mentoring
Black Cloud	• Ristrutturazione della community • Creazione di un piano di comunicazione
Lone Wolf	• Mentoring • Creazione di un piano di comunicazione • Ristrutturazione della community
Radio Silence / Bottleneck	• Mentoring • Esercizi di coesione • Creazione di un piano di comunicazione

Table 2: Soluzioni adottate per i principali community smell

Come si nota, molte delle soluzioni adottate sono trasversali e possono contribuire alla risoluzione di più di un community smell. Da queste soluzioni prende avvio l'idea di sviluppare un software che possa automatizzare il processo di mitigazione dei community smell.

5 Assunzioni

5.1 Dataset di partenza

I dataset di partenza sono stati generati con un algoritmo di generazione che utilizza algoritmi randomici per produrre un numero prefissato di dipendenti, ognuno con una serie di attributi.

Ogni riga è composta da una serie di campi, alcuni dei quali contengono valori aggregati (calcolati prima di essere inseriti all'interno del dataset, ma senza specificare le modalità di calcolo e acquisizione). Altri campi sono ridondanti e possono essere ricavati a partire dalla combinazione di alcuni attributi del dipendente.

Questa scelta si è resa necessaria per via dell'assenza di dataset reali di dipendenti di un'azienda che riportassero contemporaneamente:

- Dati legati alla collaborazione su artefatti software,
- Dati di comunicazione tra colleghi,
- Dati sui rapporti sociali tra colleghi.

Infine, la generazione dei dataset in questione è vincolata a rispettare una metrica di betweenness elevata, in modo da rispecchiare la struttura dei grafi collaborativi dei progetti open source analizzati. Sono inoltre utilizzati indici di probabilità bassi per la generazione di relazioni sociali tra dipendenti. In questo modo il dataset generato con questi vincoli si avvicina di più alla realtà, rispetto ad un dataset generato in modo completamente randomico senza vincoli particolari.

5.2 Dataset: struttura

Ogni riga rappresenta un dipendente dell'azienda che il dataset descrive; ogni colonna rappresenta un attributo (o una lista di attributi).

Il dataset descrive un'azienda con struttura gerarchica, caratterizzata da:

- un solo CEO, che fa parte del dipartimento esecutivo,
- diversi capi reparto, ognuno a capo del proprio dipartimento,
- suddivisione di ciascun dipartimento in team,
- un team che lavora su un singolo progetto software (o su una parte di esso).

Attributi del dataset (29 colonne)

Attributo	Descrizione
id	Numero identificativo del dipendente (intero).
Name	Nome del dipendente.
Gender	Genere del dipendente.
Politics	Orientamento politico del dipendente.
Religion	Orientamento religioso del dipendente.
Nationality	Nazionalità del dipendente.
Team	Identificativo del team aziendale di cui il dipendente fa parte.
Profession	Ruolo professionale svolto all'interno dell'azienda, con valore decimale che rappresenta una valutazione delle capacità.
MultipleProfessions	Lista di coppie chiave-valore che indicano i ruoli che il dipendente può ricoprire e la valutazione associata a ciascun ruolo.
Years of service	Anni di servizio all'interno dell'azienda.
Worked	Lista di colleghi con cui il dipendente ha già lavorato in precedenza.
Working	Lista di coppie chiave-valore: chiave = id dei colleghi attuali, valore = artefatto su cui collaborano.
Communicates	Lista di coppie chiave-valore: chiave = id dei colleghi con cui comunica, valore = peso (numero di canali usati).
Verbal communication details	Lista di coppie chiave-valore: comunicazioni verbali con peso di intensità.
Email communication details	Lista di coppie chiave-valore: comunicazioni via email con peso di intensità.
App messaging communication details	Lista di coppie chiave-valore: comunicazioni via app di messaggistica con peso di intensità.
Friends	Lista di coppie chiave-valore: relazioni di amicizia con peso di intensità.
Parents	Lista di coppie chiave-valore: relazioni di parentela con peso di intensità.
Relationship	Lista di coppie chiave-valore: relazioni sentimentali con peso di intensità.
Want_work_with	Lista di colleghi con cui il dipendente vorrebbe lavorare.
Social_relation	Lista di coppie chiave-valore che descrive rapporti sociali aggregati.
TeamLeader	Booleano: `true` se il dipendente è Team Leader.
DepartmentHead	Booleano: `true` se il dipendente è Capo Dipartimento.
CEO	Booleano: `true` se il dipendente è il CEO.
Departments	Nome del dipartimento di appartenenza.
TeamPurpose	Nome del team di appartenenza.
DepartmentPurpose	Nome del dipartimento di appartenenza.
DepartmentLeader	ID del Capo Dipartimento a cui il dipendente fa riferimento.
ExecutiveTeamMember	Booleano: `true` se il dipendente fa parte del team esecutivo.

6 Perfume: descrizione del tool

Perfume, a partire da un dataset contenente informazioni sull'organizzazione aziendale e sui dipendenti, crea un modello delle interazioni interne costruendo tre grafi:

- il grafo di collaborazione,
- il grafo di comunicazione,
- il grafo sociale.

Una volta creati, i grafi vengono analizzati per suggerire modifiche ottimali all'organizzazione aziendale grazie all'utilizzo di algoritmi di ottimizzazione dei grafi.

6.1 Grafi

I grafi generati da Perfume sono grafi non diretti, in cui:

- i nodi rappresentano i singoli dipendenti,

- gli archi rappresentano una relazione tra due dipendenti.

In particolare:

- Nel grafo collaborativo, un arco indica che due dipendenti lavorano sugli stessi artefatti.
- Nel grafo comunicativo, un arco descrive la comunicazione tra due dipendenti.
- Nel grafo sociale, un arco rappresenta l'intensità del legame sociale.

I grafi vengono costruiti leggendo i valori delle colonne:

- `working`, `worked` per costruire le relazione del grafo collaborativo,
- `communicates` per costruire le relazione del grafo comunicativo, ricavata come aggregazione (media aritmetica) dei dati sulle tipologie di comunicazione,
- `social_relation` per costruire le relazione del grafo sociale, ricavata come aggregazione (media aritmetica) dei dati sulle tipologie di relazioni sociali.

Una volta che i grafi sono stati costruiti ed arricchiti con tutti gli attributi dei nodi e i pesi degli archi, vengono elaborati dal risolutore che, in base ai vincoli definiti per l'ottimizzatore e alle istruzioni sui community smells ad esso affidate, cercherà una soluzione per ottimizzare i grafi in modo da limitare la comparsa dei community smells.

6.2 Resolver

Il *Resolver* è il cuore di Perfume. Analizza i tre grafi costruiti dal dataset e produce suggerimenti di modifiche volte a ridurre la probabilità di insorgenza di Community Smells.

Questo componente è formato da due algoritmi:

- **Silo & Wolf Resolver**,
- **Bottleneck & Cloud Resolver**,

che applicano le strategie risolutive proposte in letteratura.

Ottimizzatore Per la fase di ottimizzazione è stato impiegato **Gurobi**, un solutore di programmazione matematica in grado di generare, a partire da un modello formulato mediante vincoli e variabili decisionali, le soluzioni ottimali (o prossime all'ottimo) rispetto all'obiettivo definito.

6.3 Silo & Lone Wolf Resolver

6.3.1 Silos Organizzativi

I Silos organizzativi sono quei community smell in cui dei team formano dei silos. Questa caratterizzazione non permette ai membri del team di comunicare al di fuori del team stesso e porta al peggioramento dello scambio di informazioni.

Come riportato nel paper *Refactoring Community Smells in the Wild: The Practitioner's Field Manual*, sono state suggerite le seguenti soluzioni:

- Restructure the community (20%)
- Create communication plan (20%)
- Mentoring (20%)

Prendendo come spunto queste ricerche, si è voluto implementare una soluzione che prenda in considerazione il refactoring e il mentoring.

L'implementazione si pone l'obiettivo di risolvere i silos ristrutturando i team e i reparti dell'azienda, secondo alcune assunzioni.

Assunzioni

1. Due team si comunicheranno più volentieri se i nodi dei team in questione hanno un certo grado di relazione sociale.
2. Le persone con anni di esperienza, individuate come *mentori*, possono non solo contrastare lo smell *lone wolf*, ma anche rompere i silos, poiché sanno come comunicare e favorire la collaborazione.
3. Se due persone hanno già lavorato assieme, sarà più facile per una di loro chiedere consigli e aiuto all'altra.

La soluzione implementativa utilizza quindi queste assunzioni per cambiare e ristrutturare il grafo in modo tale che il community smell silos possa risolversi.

Possiamo avere una combinazione di queste assunzioni per ottenere una certa intensità o potenza nella soluzione. Infatti, possiamo vedere le assunzioni come variabili e combinarle insieme:

amici	mentore	hanno lavorato insieme
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Dalla tabella viene rimossa la combinazione 000. Inoltre, pur avendo considerato tale strategia nella soluzione proposta, essa è stata semplificata nell'implementazione omettendo la popolazione della relazione *worked*, che costituiva l'unico elemento in grado di rappresentare lo stato precedente del sistema.

6.3.2 Lone Wolf

Attraverso la soluzione offerta per i silos organizzativi vi è l'opportunità anche di affrontare lo smell *lone wolf*. Secondo le ricerche, le soluzioni che i practitioner hanno applicato con successo sono:

- Mentoring (42%)
- Restructure the community

Nella soluzione dei silos possiamo dunque integrare la gestione dei *lone wolf*. Una volta identificati questi elementi, possiamo affiancarvi dei mentori, gli stessi che sono stati distribuiti nei vari team durante il refactoring.

L'idea è quindi di aggiungere, durante l'ottimizzazione, un nuovo tipo di arco che congiunge i possibili lone wolf ai mentori, in modo tale da ridurre la probabilità di questo smell.

6.3.3 Modello di Ottimizzazione

Variabili di decisione

$$x_{i,t} = \begin{cases} 1 & \text{se il dipendente } i \in \mathcal{E} \text{ è assegnato al team } t \in \mathcal{T} \\ 0 & \text{altrimenti} \end{cases}$$

$$y_t = \begin{cases} 1 & \text{se il team } t \in \mathcal{T} \text{ è attivo (almeno un membro)} \\ 0 & \text{altrimenti} \end{cases}$$

$$z_t = \begin{cases} 1 & \text{se il team } t \in \mathcal{T} \text{ contiene almeno un senior} \\ 0 & \text{altrimenti} \end{cases}$$

Funzione obiettivo Si massimizza la combinazione pesata di quattro componenti:

$$\max \underbrace{\sum_{(i,j)\in\mathcal{F}} w_{ij}\left(1-\sum_{t\in\mathcal{T}} x_{i,t}x_{j,t}\right)}_{\text{Separazione legami di amicizia}}$$

$$+ \underbrace{\lambda \sum_{(i,j)\in\mathcal{B}}\sum_{t\in\mathcal{T}} x_{i,t}x_{j,t}}_{\text{Raggruppamento su artefatti correlati}}$$

$$+ \underbrace{\beta \sum_{(i,j)\in\mathcal{W}}\sum_{t\in\mathcal{T}} x_{i,t}x_{j,t}}_{\text{Preferenze ``want\_work\_with''}}$$

$$+ \underbrace{\gamma \sum_{t\in\mathcal{T}} z_t}_{\text{Bonus presenza senior}}$$

Dove: - $\mathcal{F}$ = insieme delle coppie di dipendenti legati da amicizia con peso w_{ij} - $\mathcal{B}$ = insieme delle coppie di dipendenti legati da artefatti correlati - $\mathcal{W}$ = insieme delle coppie di dipendenti con preferenza esplicita a lavorare insieme - λ, β, γ = parametri di peso per i diversi contributi

Vincoli

1. **Assegnazione univoca**
$$\sum_{t\in\mathcal{T}} x_{i,t} = 1 \quad \forall i \in \mathcal{E}$$
2. **Cardinalità team**
$$\cdot y_t \;\le\; \sum_{i\in\mathcal{E}} x_{i,t} \;\le\; \cdot y_t \quad \forall t \in \mathcal{T}$$
3. **Attivazione team**
$$x_{i,t} \le y_t \quad \forall i \in \mathcal{E},\ \forall t \in \mathcal{T}$$
4. **Numero di team attivi**
$$\le \sum_{t\in\mathcal{T}} y_t \le$$
5. **Eterogeneità professionale**
$$\sum_{\substack{i\in\mathcal{E}\\ p(i)=p}} x_{i,t} \;\le\; M_p \quad \forall t \in \mathcal{T},\ \forall p \in \mathcal{P}$$
dove $p(i)$ è la professione del dipendente i, e M_p è il massimo consentito di membri per professione p.
6. **Presenza senior**
$$\sum_{\substack{i\in\mathcal{E}\\ s(i)=1}} x_{i,t} \;\ge\; z_t \quad \forall t \in \mathcal{T}$$
$$z_t \le y_t \quad \forall t \in \mathcal{T}$$
7. **Separazione amicizie forti**
$$x_{u,t} + x_{v,t} \le 1 \quad \forall (u,v) \in \mathcal{F} : w_{uv} > \theta_{\text{social}},\ \forall t \in \mathcal{T}$$

Obiettivo Strategico e Impatto sulla Riduzione dei Silos Massimizzando la funzione obiettivo, il modello mira a ottenere team che:

- Promuovono la collaborazione interna,
- Rafforzano la comunicazione intra-team,
- Gestiscono i legami personali stretti tramite penalità, anziché proibizioni assolute,
- Permettono al risolutore di determinare il numero ottimale di team.

Questo approccio strategico punta a creare team coesi e ben integrati, riducendo il rischio di silos organizzativi e migliorando la produttività complessiva.

6.4 Bottleneck & Cloud Resolver

Con il presente *resolver* ci poniamo l'obiettivo non solo di affrontare eventuali *bottleneck*, ma anche di proporre, mediante l'instaurazione di un piano di comunicazione, una possibile soluzione al fenomeno del *black cloud*.

Il riconoscimento di questo *Community Smell* richiede un'analisi dei cluster all'interno del grafo: se due cluster risultano scarsamente connessi tra loro, è possibile che si configuri un *Bottleneck*. In particolare, la presenza di un membro di una sotto-comunità che funge da unico intermediario nella comunicazione con due o più altre sotto-comunità rappresenta una situazione tipica di *Bottleneck*, soprattutto in ambito comunicativo.

L'approccio adottato è finalizzato all'individuazione dei cosiddetti *broker* all'interno del grafo, ossia quei nodi che, pur appartenendo a un determinato cluster, instaurano collegamenti con cluster differenti e diventano, di fatto, gli unici punti di passaggio in specifici percorsi di comunicazione.

6.4.1 Modello di Ottimizzazione

Il codice Python fornito implementa una metodologia in più fasi per risolvere un problema di ottimizzazione volto a mitigare i bottleneck di comunicazione all'interno di un'organizzazione.

L'approccio combina l'analisi delle reti sociali (con i grafi di comunicazione, lavorativo e sociale) con un modello di programmazione lineare intera (Integer Linear Programming, ILP).

Il problema è formulato come una massimizzazione del beneficio, che deriva dall'aggiunta di nuovi canali di comunicazione e dalla promozione di nodi chiave, tenendo conto dei costi associati.

Definizione del Problema e delle Variabili di Decisione Sia V l'insieme dei dipendenti (nodi), con $|V| = n$. Vengono considerati tre grafi diretti:

- $G_C = (V, E_C, w_C)$: Il grafo di comunicazione, dove un arco $(i, j) \in E_C$ indica un canale di comunicazione esistente tra i nodi i e j, con un peso w_{ij} che rappresenta l'intensità della comunicazione.
- $G_L = (V, E_L, w_L)$: Il grafo lavorativo (o delle preferenze), dove un arco $(i, j) \in E_L$ indica una relazione lavorativa, con un peso w_{ij} che rappresenta la preferenza lavorativa.
- $G_S = (V, E_S, w_S)$: Il grafo sociale, dove un arco $(i, j) \in E_S$ indica una relazione sociale, con un peso w_{ij}.

L'obiettivo è scegliere un insieme di nuovi archi da aggiungere a G_C e un insieme di nodi da "promuovere" per massimizzare una funzione di beneficio netto.

Le variabili di decisione sono:

- $z_{ij} \in \{0, 1\}$ per ogni coppia di nodi (i, j) con $i \neq j$. $z_{ij} = 1$ se viene aggiunto un nuovo arco da i a j. Per i grafi non diretti, si impone la simmetria $z_{ij} = z_{ji}$.
- $p_k \in \{0, 1\}$ per ogni nodo $k \in V$. $p_k = 1$ se il nodo k viene promosso a intermediario.

Funzione Obiettivo: Massimizzazione del Beneficio Netto La funzione obiettivo Z è una combinazione lineare di tre termini principali: il beneficio degli archi aggiunti, il beneficio dei nodi promossi e i costi associati.

$$\text{Massimizza } Z = \left(\sum_{i \in V} \sum_{j \in V, i \neq j} B_{ij} z_{ij} \right) + \left(\sum_{k \in V} P_k p_k \right) - \left(C_E \sum_{i,j} z_{ij} + C_P \sum_{k} p_k \right)$$

Dove:

- * B_{ij}: Il beneficio dell'aggiunta dell'arco (i, j). Questo beneficio è calcolato in modo euristico e multifattoriale:
  - Beneficio di connessione tra comunità: Se l'arco (i, j) connette due comunità problematiche identificate come bottleneck, un bonus significativo viene aggiunto, proporzionale a un "punteggio di rischio" risk_{ij} associato alla coppia di comunità.
  - Qualità dei nodi: Un bonus viene aggiunto in base alla qualità dei nodi i e j, misurata dalla loro centrality (es. degree centrality) nei grafi lavorativo e sociale. Si utilizza un fattore di peso Weight_Node_Quality.
  - P_k: Il valore di promozione del nodo k. Questo valore dipende dal ruolo del nodo k:
    - Un valore maggiore viene assegnato se k è stato identificato come un bottleneck unificato (sia comunicativo che lavorativo).
    - Un valore aggiuntivo viene assegnato se k è un influencer sociale.

* Ulteriori bonus vengono assegnati se la promozione di k "sblocca" la possibilità di collegare comunità problematiche tramite ponti sociali esistenti.
* C_E: Il costo per l'aggiunta di un singolo arco.
* C_P: Il costo per la promozione di un singolo nodo.

Vincoli Il modello di ottimizzazione è soggetto a diversi vincoli che ne definiscono lo spazio di soluzioni ammissibili:

- **Limite massimo di nuovi archi totali**: Il numero totale di nuovi archi non può superare un valore predefinito K_{max_edges}.
$$\sum_{i \in V} \sum_{j \in V, i \neq j} z_{ij} \leq K_{max\_edges}$$
Per i grafi non diretti, la somma è sulla metà superiore della matrice di adiacenza per evitare il doppio conteggio.

- **Limite massimo di nuovi archi per ogni bottleneck**: Per ogni nodo k identificato come un bottleneck unificato, il numero di archi incidenti (aggiunti) è limitato da $K_{max\_edges\_per\_bottleneck}$.
$$\sum_{j \in V, j \neq k} (z_{kj} + z_{jk}) \leq K_{max\_edges\_per\_bottleneck} \quad \forall k \in V_{\text{unified\_bottlenecks}}$$

- **Limite massimo di nodi promossi**: Il numero totale di nodi promossi non può superare un valore predefinito $P_{max\_promoted}$.
$$\sum_{k \in V} p_k \leq P_{max\_promoted}$$

- **Limite massimo di nodi promossi**: Il numero totale di nodi promossi non può superare un valore predefinito $P_{max\_promoted}$.
$$\sum_{k \in V} p_k \leq P_{max\_promoted}$$

- **Contribuzione dei nodi promossi**: Se un nodo k viene promosso, deve contribuire alla creazione di un numero minimo di nuovi archi, $M_{min\_new\_edges}$.
$$\sum_{j \in V, j \neq k} (z_{kj} + z_{jk}) \geq M_{min\_new\_edges} \cdot p_k \quad \forall k \in V$$

- **Vincolo di esclusione**: Un nodo identificato come un bottleneck unificato non può essere promosso a intermediario.
$$p_k = 0 \quad \forall k \in V_{\text{unified\_bottlenecks}}$$

- **Vincoli impliciti**: * Gli archi esistenti non possono essere aggiunti di nuovo: se $(i, j) \in E_C$, allora $z_{ij} = 0$. * Non sono permessi self-loops: $z_{ii} = 0$.

Algoritmo di Risoluzione e Post-Processing Il problema è risolto utilizzando il solutore **Gurobi** che implementa algoritmi di branch-and-cut per la programmazione lineare intera.

Dopo aver trovato la soluzione ottimale, il codice esegue un'ulteriore fase di post-processing per generare un "piano di comunicazione". Questo piano è rappresentato da un **albero ricoprente massimo (Maximum Spanning Tree, MST)**.

- Viene creato un nuovo grafo $G_{plan\_candidate}$ che include tutti i nodi.
- Gli archi esistenti in G_C vengono aggiunti a $G_{plan\_candidate}$ con un peso basso (o nullo).
- Gli archi (i, j) scelti dal modello ILP (dove $z_{ij} = 1$) vengono aggiunti a $G_{plan\_candidate}$ con un peso molto elevato per garantire che siano inclusi nell'MST.
- L'algoritmo di Kruskal o Prim viene eseguito su $G_{plan\_candidate}$ per trovare l'MST, che rappresenta il piano di comunicazione ottimale per connettere i dipendenti con il minor numero di archi ma massimizzando i "ponti" più importanti.

Il risultato finale non è solo il grafo di comunicazione migliorato, ma anche un piano d'azione specifico (l'MST) che elenca gli archi essenziali da stabilire per migliorare la connettività globale e mitigare i bottleneck.

7 Risultati

In questa sezione vengono presentati i risultati più rilevanti dell'elaborazione di Perfume, per capire sia come esso si comporta durante la risoluzione dei modelli, sia quali sono in potenza gli effetti dei cambiamenti che esso introduce nell'organizzazione aziendale. Prima viene descritta la pipeline che forma il modulo che sintetizza le statistiche, poi vengono presentati i test effettuati.

Statistiche Per formulare delle statistiche coerenti con i modelli matematici ottimizzati dai risolutori, cioè dei grafi, si è scelto di analizzare sia alcune metriche provenienti dall'analisi delle reti sociali, sia di riconoscere i motivi (motifs) prima e dopo l'ottimizzazione, per confrontarne le distribuzioni.

7.1 Metriche

Per comprendere l'entità dei miglioramenti inseriti da Perfume all'interno dell'organizzazione in esame, vengono misurate 4 metriche di centralità sui grafi che la modellano, prima e dopo l'ottimizzazione. Le metriche, che vengono mostrate sotto forma di distribuzioni, sono: degree centrality, betweenness centrality, closeness centrality ed eigenvector centrality. Le metriche citate quantificano la centralità di un nodo all'interno della rete rispetto a specifiche caratteristiche:

Degree Il grado di un nodo è una misura di centralità che indica quanto quel nodo è connesso ad altri nodi della rete, in un grafo non diretto e non pesato il grado di un nodo è il numero di archi connessi a quel nodo. La distribuzione di grado permette di capire quanti nodi all'interno del grafo sono prevalentemente isolati e quanti sono connessi con molti altri nodi.

$$C_D(v) = deg(v)$$

Betweenness Misura di centralità basata su percorsi, cioè sulla distanza media, la betweenness rappresenta la capacità di un nodo di fungere da intermediario nel connettere altri nodi della rete all'interno di percorsi a distanza minima, cioè lungo le geodetiche. Quindi descrive quali nodi sono più propensi a favorire il passaggio di informazioni all'interno del grafo, oppure quali nodi hanno un maggiore controllo nella rete di diffusione di informazioni.

$$C_B(v) = \sum_{i \neq j, i \neq v, j \neq v} \frac{\sigma_{ij}(v)}{\sigma_{ij}}$$

Closeness Misura di centralità basata su percorsi, cioè sulla distanza media, misura la vicinanza del nodo i rispetto a tutti gli altri nodi della rete. La closeness di un nodo è tanto più elevata quanto minore è la distanza media di quel nodo dagli altri nodi della rete. Descrive quali sono i nodi più semplicemente raggiungibili a partire dal nodo in esame.

$$C_C(v) = \frac{1}{\sum_{i \in V} d(i, v)}$$

7.2 Motif analisys

Un motivo è un gruppo di elementi proveniente dallo stesso insieme. Se a questa definizione si applicano delle restrizioni topologiche, si possono utilizzare i motivi per ricercare strutture simili tra di loro all'interno di un grafo, da cui la definizione per la teoria dei grafi: un motivo è un sotto-grafo o uno specifico pattern di connettività che compare frequentemente all'interno di un grafo più grande. La motif analisys è proprio la ricerca e l'analisi dei motivi ricorrenti all'interno dei grafi.

Networkit Il tool utilizzato per ricercare motivi all'interno di un grafo è Networkit, un modulo Python per l'analisi di reti che implementa algoritmi efficienti che sfruttano computazione parallela su architetture multi-core. Non solo è adatto a calcolare metriche sui grafi e a cercare strutture ricorrenti al loro interno, ma è anche compatibile con NetworkX, cioè la libreria Python utilizzata per modellare un grafo.

Il modulo di motif analisys è stato progettato per ricercare 3 tipologie di motivi: cliques, stars e barbells.

Cliques NetworKit implementa una variante dell'algoritmo di Bron–Kerbosch, scelto perchè è l'algoritmo più veloce, per restituire tutte le Cliques(cricche) massimali senza ridondanze. 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. La dimensione di una cricca è definita come il numero di vertici che contiene.

Stars Una stella di dimensione k è composta da un nodo da cui escono k archi, ognuno verso un altro nodo differente da quello di partenza e dagli altri che formano la stella. Per ogni nodo calcola prima il grado di ogni nodo, poi, se il grado è maggiore o uguale della dimensione della stella meno uno, conta con quante combinazioni di nodi vicini il nodo corrente può formare una stella di dimensione k:

$$N_s = \binom{deg}{k-1} \qquad per \qquad deg >= k-1$$

Barbells Una barbell (bilanciere) è una coppia di cricche massimali collegate da un arco o da un solo percorso che passa per 1 o più nodi. L'algoritmo di ricerca delle barbells utilizza l'algoritmo di Bron–Kerbosch, fornito da NetworKit, per trovare le cricche massimali. L'insieme di cricche trovate vengono filtrate per eliminare le sotto-cricche ridondanti. A questo punto, per ogni coppia di cricche:

- verifica che non abbiano troppi nodi in comune,
- per ciascun nodo in clique1, esegue una BFS (Breadth-First Search) per trovare percorsi verso clique2,
- controlla che il cammino trovato abbia una lunghezza inferiore alla lunghezza massima impostata come parametro dall'utente e che non attraversi uno dei nodi appartenenti a una delle due cricche in esame.

Anche i motivi vengono mostrati e confrontati attraverso distribuzioni.

7.3 Test

In questo paragrafo vengono presentati alcuni test per mostrare come è stato misurato il cambiamento introdotto dall'ottimizzazione di Perfume. I dataset di input sono stati prima generati con un algoritmo genetico pseudo-randomico, prendento come punto di partenza le metriche calcolate su dei grafi collaborativi di progetti reali. Poi sono stati introdotti dei nodi aggiuntivi che erano disposti a formare un community smells, come specificato dalla letteratura.
I test consistono in:

- specificare l'input per Perfume in formato csv,
- eseguire il tool impostando i coefficienti dei vincoli,
- calcolare le metriche dei grafi di partenza e di quelli ottimizzati,
- ricercare i motivi sia nei grafi di partenza, sia in quelli ottimizzati,
- confrontare i risultati ottenuti.

Il testing è stato effettuato per il *Bottelneck Resolver* e per entrambe le versioni implementate del *Silo Resolver*, inoltre ogni test è stato ripetuto per un dataset di input di diversa grandezza.

Test Silo Resolver 1.1 - 42 dipendenti Il csv di partenza è di 42 dipendenti, quindi i grafi avranno 42 nodi.
Di seguito i vincoli scelti per l'ottimizzazione:

min_teams: 1,

max_teams: 15,

min_team_size: 1,

max_team_size:20,

max_members_per_profession:10,

theta_social:1.0,

theta_senior:30.0,

lone_wolf_comms_threshold:0.30,

lambda_bipartite:1.0,

bonus_want_work_with:1.0,

bonus_senior_presence:1.0.

Di seguito i parametri di Gurobi per questo test:

TimeLimit: 600,

MIPGap: 0.0,

OutputFlag: 1,

MIPFocus:0,

Heuristics:0.05,

Cuts:1.

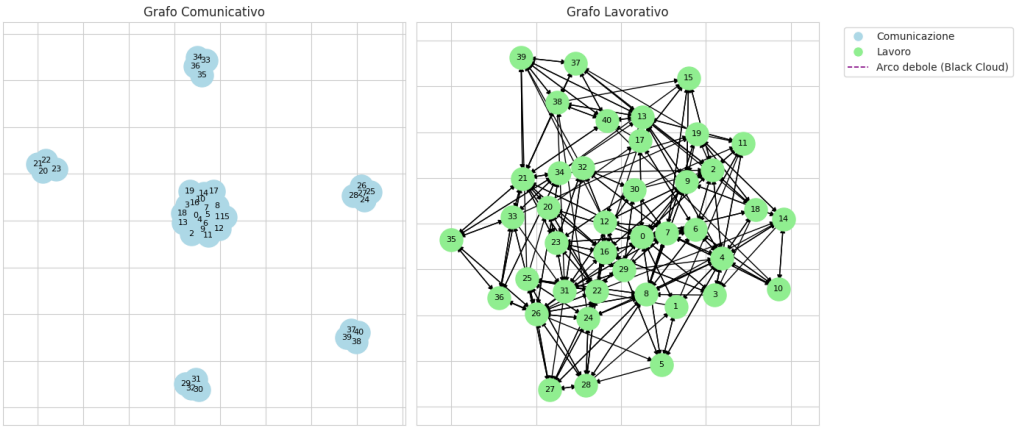

Figure 1: Grafi Comunicativo e Collaborativo PRE ottimizzazione.

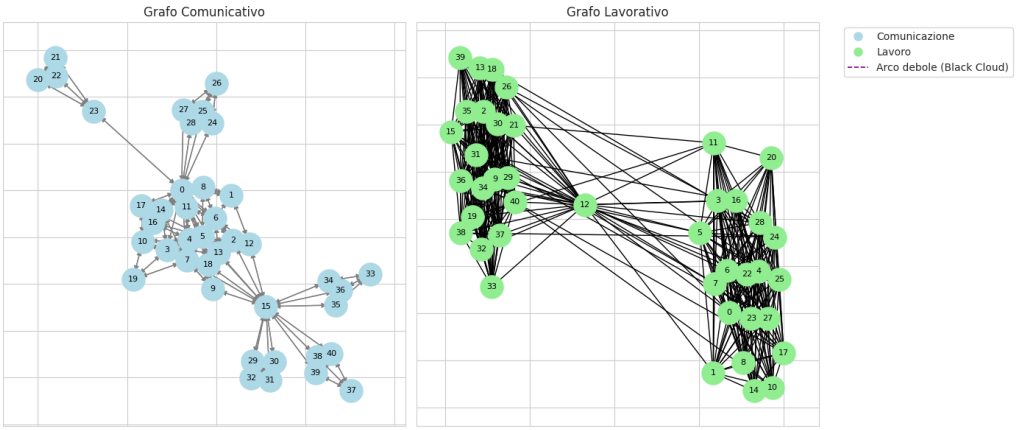

Figure 2: Grafi Comunicativo e Collaborativo POST ottimizzazione.

Dalle distribuzioni di closeness del grafo collaborativo, misurate prima e dopo l'ottimizzazione, si può vedere come, nella seconda distribuzione, la closeness massima è aumentata, e sono aumentati anche i nodi con closeness > 0.35. Quindi, nel complesso, i nodi del grafo comunicativo ottimizzato sono più vicini, ciò vuol dire che la comunicazione tra di essi è più efficiente.

Ciò è visibile anche dal miglioramento che il grafo comunicativo ottiene dal processo di ottimizzazione, che mostra i silo organizzativi aggregarsi alla parte più grande del grafo di partenza, e quindi essere più "vicini" agli altri nodi del grafo.

Per quanto riguarda il grafo collaborativo, l'ottimizzazione produce sempre la stessa soluzione per tutti i differenti dataset analizzati, che è la soluzione in cui tutti i nodi sono connessi con tutti, che massimizza le metriche delle reti sociali, come la distribuzione di grado, la closeness, ma non è una soluzione fattibile nella realtà. Questo limite di Perfume è descritto meglio nella prossima sezione.

Entrambi i grafi presentano un generale aumento di motivi come cricche e stelle, come risultato dell'aggiunta

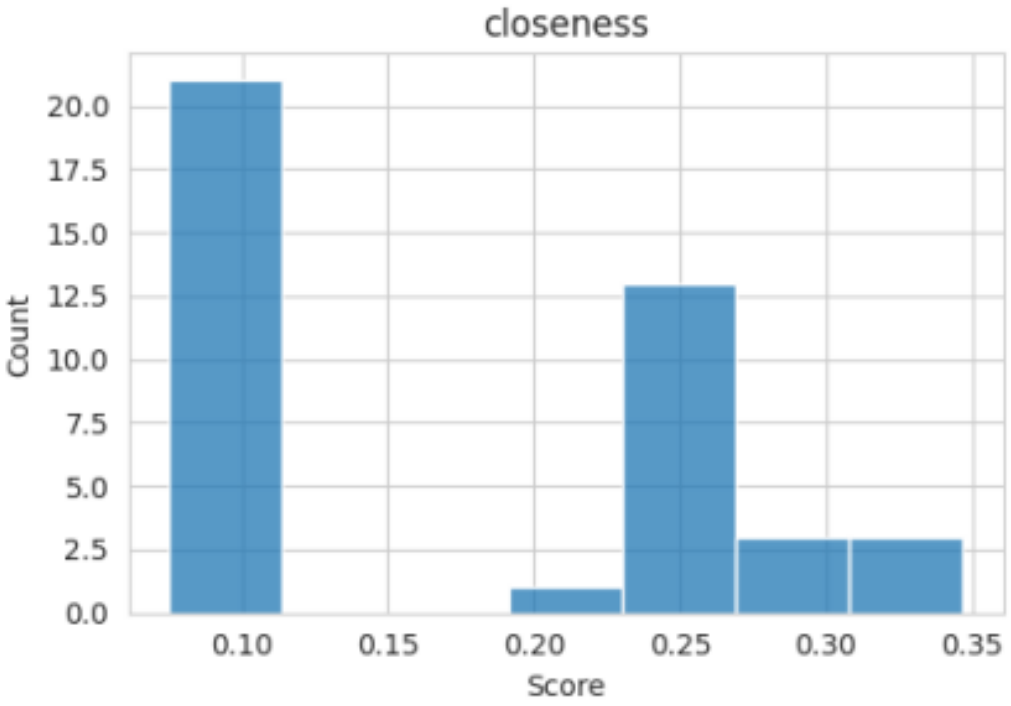

Figure 3: Distribuzione di Closeness del grafo Comunicativo PRIMA dell'ottimizzazione.

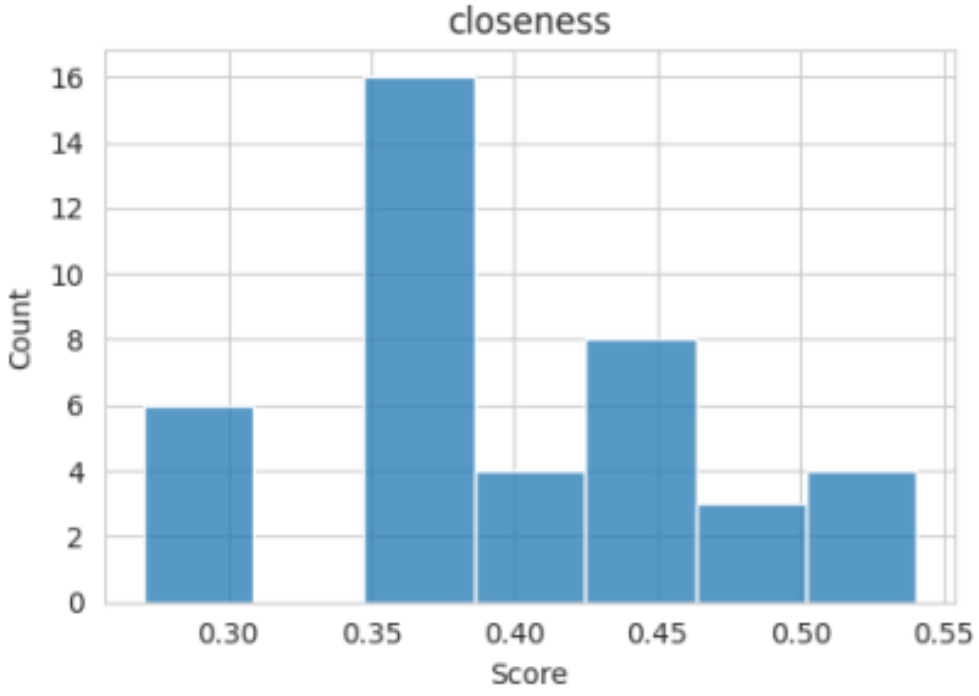

Figure 4: Distribuzione di Closeness del grafo Comunicativo DOPO dell'ottimizzazione.

dei nuovi archi da parte dell'ottimizzatore. Infine, il grafo comunicativo presenta la comparsa di barbells dovuta alla connessione delle cricche formate dai silo organizzativi con il resto del grafo comunicativo.

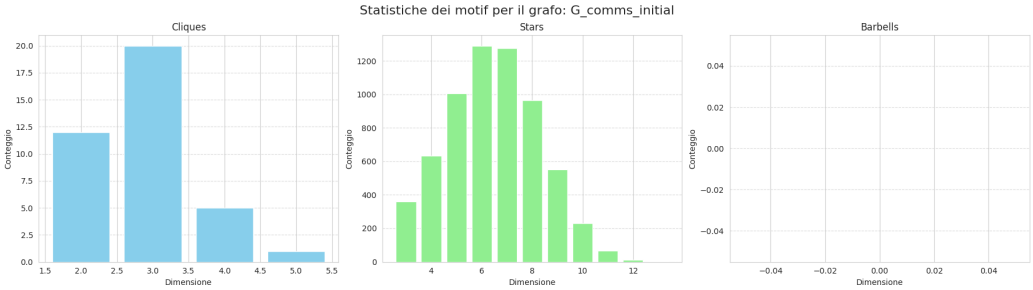

Figure 5: Distribuzione di motivi nel grafo Comunicativo PRIMA dell'ottimizzazione.

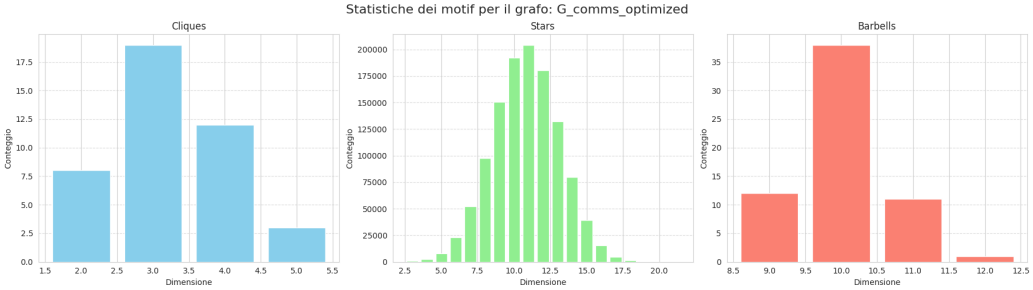

Figure 6: Distribuzione di motivi nel grafo Comunicativo DOPO dell'ottimizzazione.

Test Silo Resolver 1.1 - 121 dipendenti Il csv di partenza è di 121 dipendenti, quindi i grafi avranno 121 nodi. I vincoli e i parametri di Gurobi sono identici a quelli del test precedente.
In questo caso la closeness resta più o meno la stessa, aumenta solo per i nodi che vengono collegati al cluster

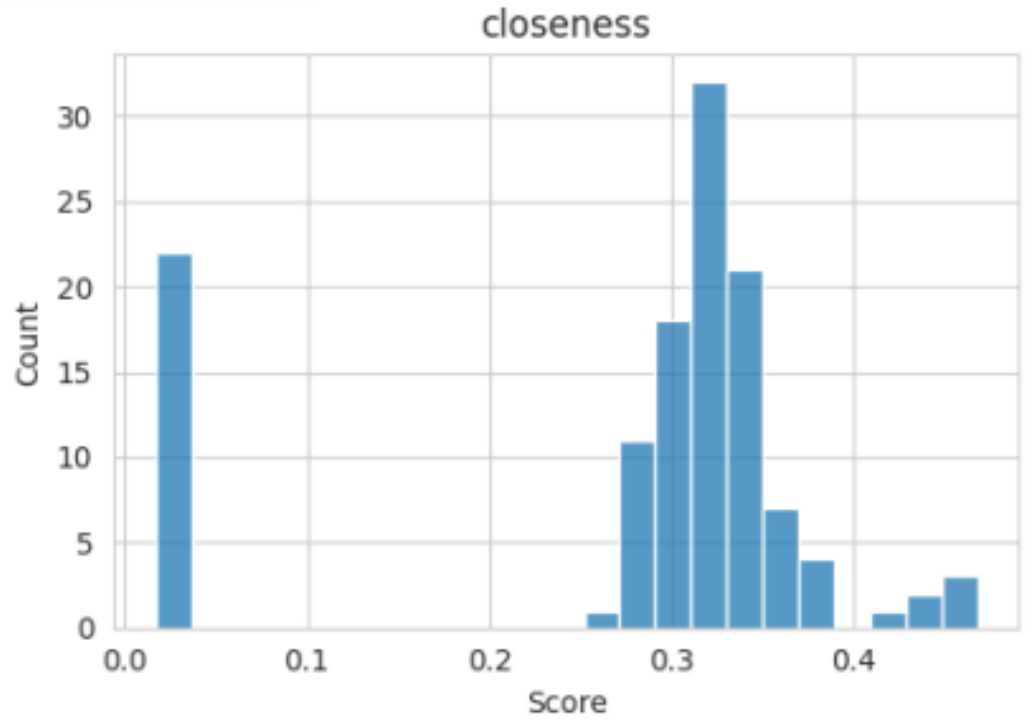

Figure 7: Distribuzione di Closeness del grafo Comunicativo PRIMA dell'ottimizzazione.

più grande del grafo. Anche in questo caso i motivi aumentano tutti di numero e di dimensione, confermando il fatto che, sia nel grafo comunicativo che in quello collaborativo vengono inserite nuove connessioni.

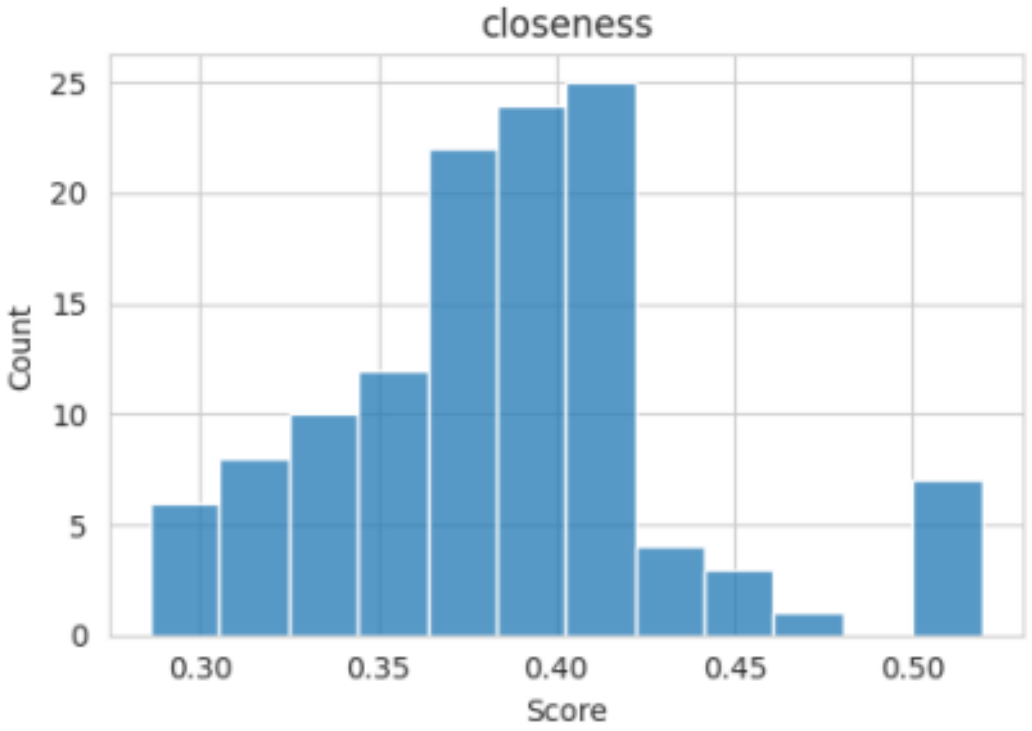

Figure 8: Distribuzione di Closeness del grafo Comunicativo DOPO dell'ottimizzazione.

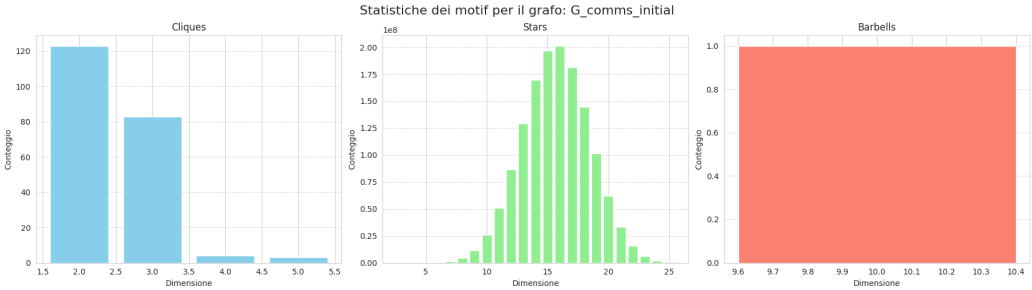

Figure 9: Distribuzione di motivi nel grafo Comunicativo PRIMA dell'ottimizzazione.

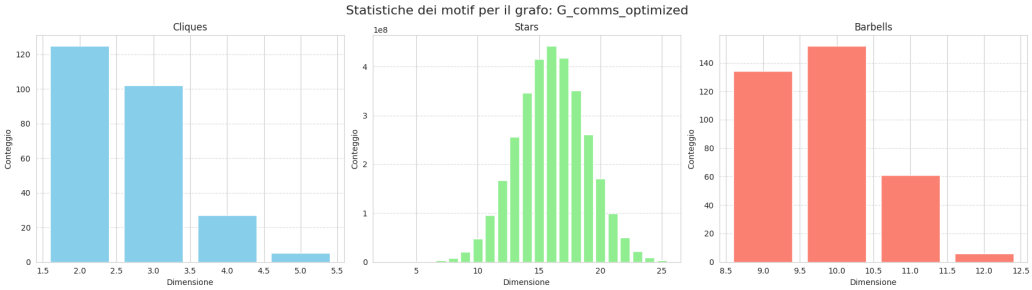

Figure 10: Distribuzione di motivi nel grafo Comunicativo DOPO dell'ottimizzazione.

Test Silo Resolver 1.2 - 81 dipendenti Il csv di partenza è di 81 dipendenti, quindi i grafi avranno 81 nodi.
Di seguito i vincoli scelti per l'ottimizzazione:

min_teams: 1,

max_teams: 9,

min_team_size: 1,

max_team_size:10,

max_members_per_profession:10,

theta_social:1.0,

theta_senior:30.0,

lone_wolf_comms_threshold:0.30,

lambda_bipartite:1.0,

bonus_want_work_with:1.0,

bonus_senior_presence:1.0.

Di seguito i parametri di Gurobi per questo test:

TimeLimit: 600,

MIPGap: 0.0,

OutputFlag: 1,

MIPFocus:0,

Heuristics:0.05,

Cuts:1.

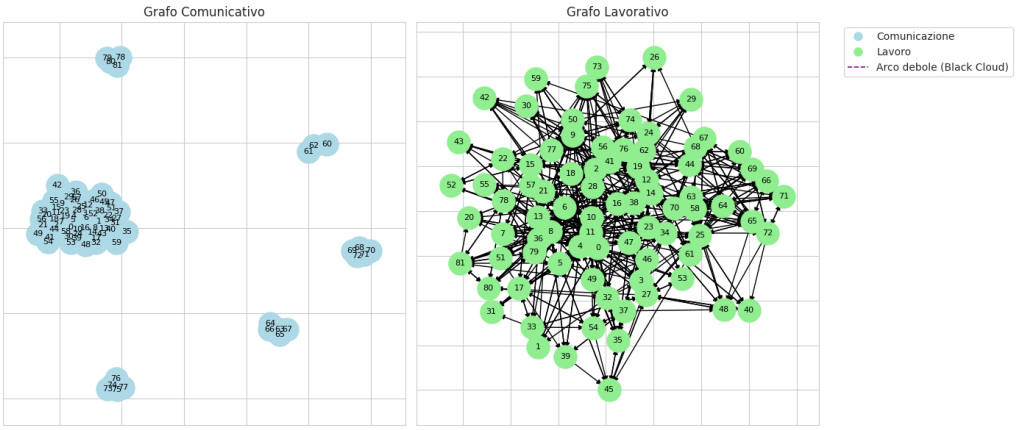

Figure 11: Grafi Comunicativo e Collaborativo PRE ottimizzazione.

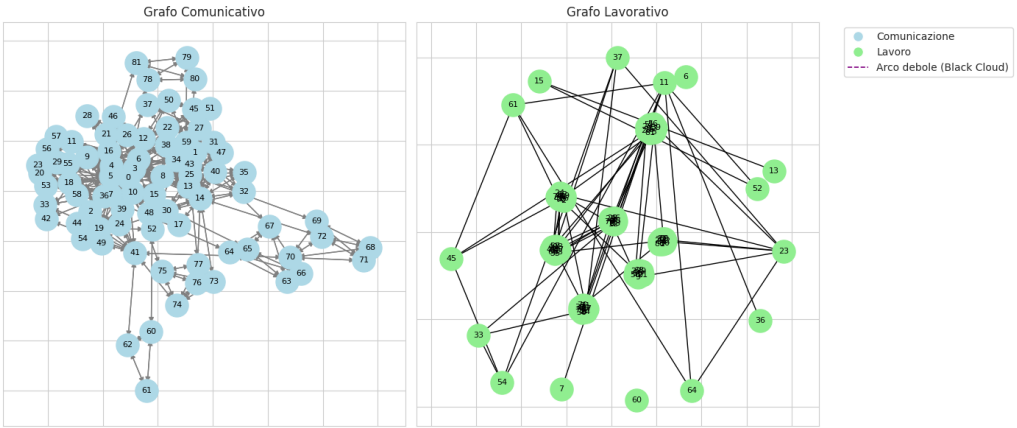

Figure 12: Grafi Comunicativo e Collaborativo POST ottimizzazione.

Avendo inserito il termine di penalità nella funzione obiettivo, che viene conteggiata per ogni operazione che allontana il grafo ottimizzato da quello di partenza, l'aggiunta di nuovi archi è scoraggiata e il resolver è forzato

a ricercare tra le soluzioni più vicine alla rete di partenza. In questo modo il grafo ottimizzato non sarà più la semplice connessione di ogni nodo con tutti gli altri, ma una configurazione del grafo più verosimile, seppur ricca di connessioni(per questo grafo sono quasi 400 archi).
Il limite di questa soluzione sono però le metriche, tra cui la closeness aumenta con lo stesso andamento (per il grafo comunicativo) ma le altre sono meno confortanti, con la differenza che i team vengono ricostruiti totalmente, come avviene anche nella versione 1.1 del Silo Resolver, tenendo conto delle relazioni sociali. Ciò, come spiegato nell'ultima sezione, potrebbe essere utile applicando il metodo in modo periodico nel tempo.
Per quanto riguarda il grafo collaborativo, la distribuzione di grado dei nodi si concentra in un intervallo più

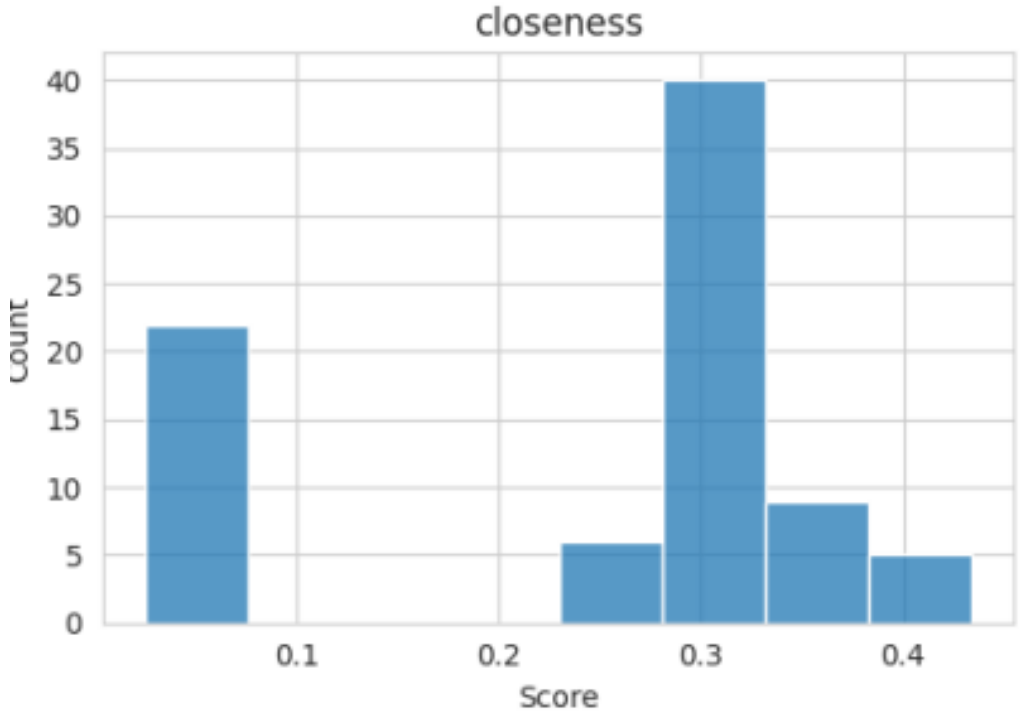

Figure 13: Distribuzione di Closeness del grafo Comunicativo PRIMA dell'ottimizzazione.

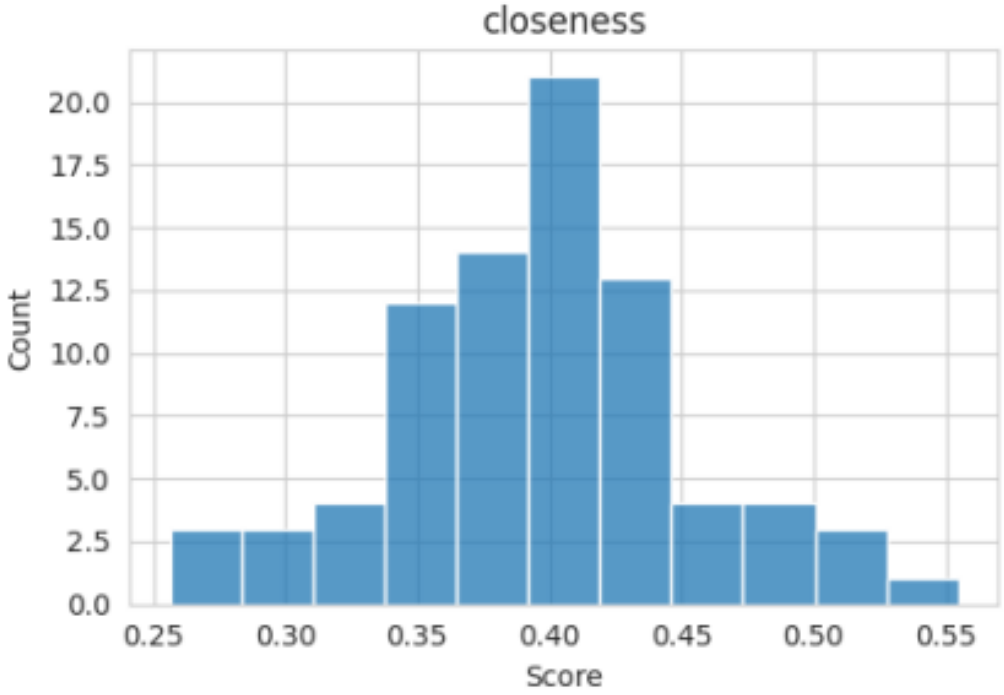

Figure 14: Distribuzione di Closeness del grafo Comunicativo DOPO dell'ottimizzazione.

piccolo dopo l'ottimizzazione (27 nodi hanno centralità di grado pari a 9, mentre nel grafo non ottimizzato al massimo 7 nodi avevano la stessa centralità di grado, con la presenza di nodi con anche 40 archi). Questo è un indizio che l'ottimizzatore ha distribuito meglio il lavoro dei dipendenti sugli artefatti, con una simile allocazione di risorse umane per ogni artefatto software, evitando che un singolo dipendente si occupi si troppi artefatti software. Per quanto riguarda l'analisi dei motivi, l'ottimizzazione del grafo comunicativo comporta l'aumento

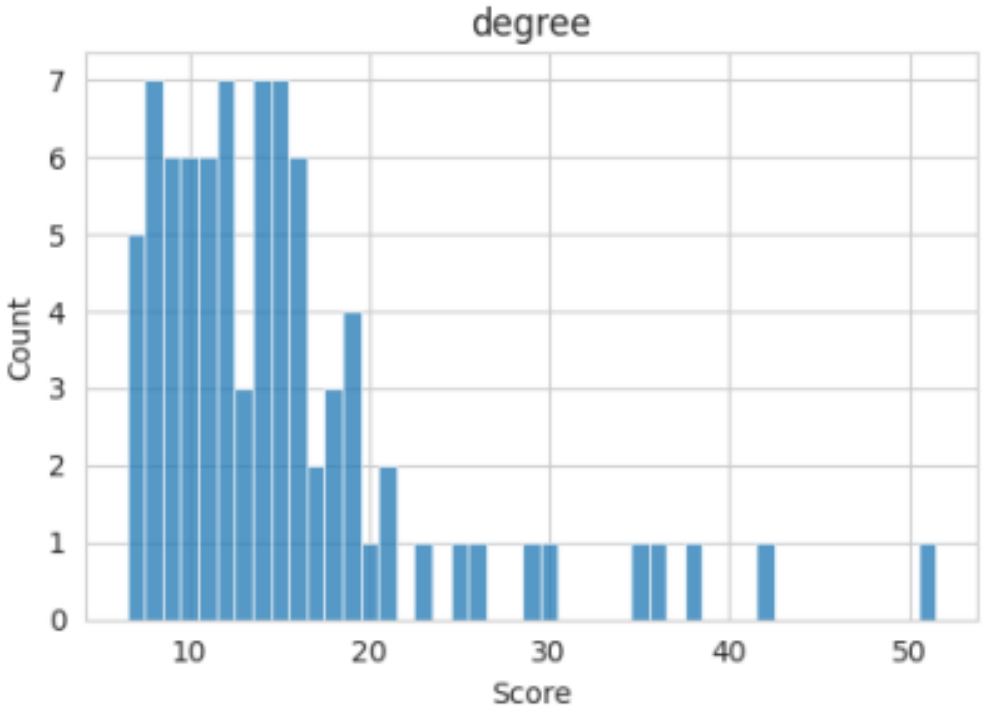

Figure 15: Distribuzione di Grado del grafo Collaborativo PRIMA dell'ottimizzazione.

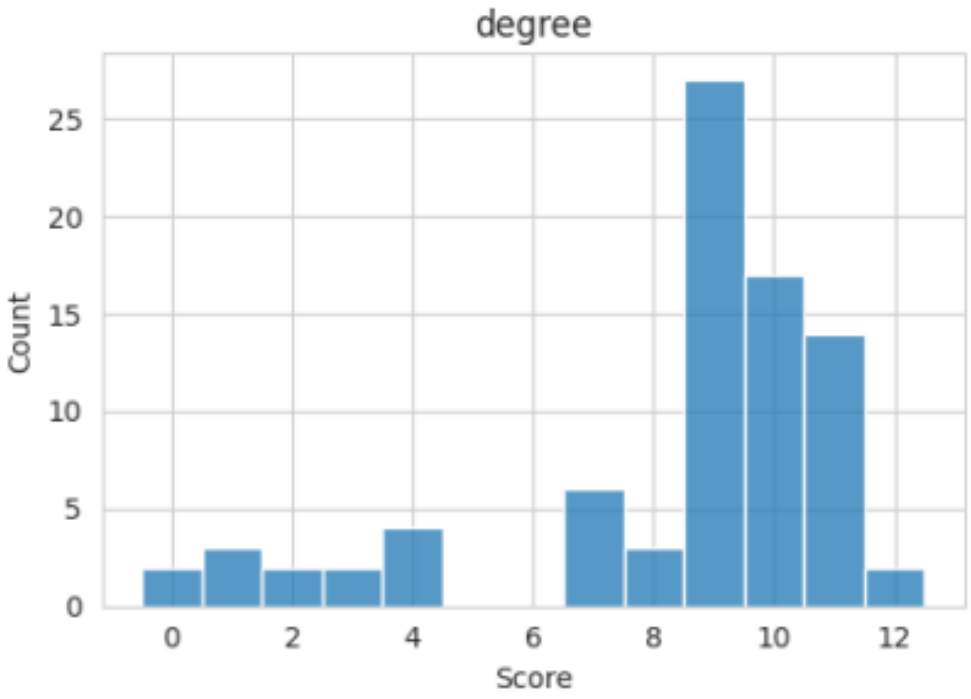

Figure 16: Distribuzione di Grado del grafo Collavorativo DOPO dell'ottimizzazione.

di tutti i motivi analizzati, con la comparsa di barbells che derivano dalla connessione delle cricche isolate al cluster più grande del grafo. L'ottimizzazione del grafo collaborativo, invece, vede una riduzione di tutti i motivi dopo l'ottimizzazione. Ciò deriva sicuramente dalla ricomposizione dei team, i cui componenti hanno il giusto

carico di lavoro e, avendo tenuto conto della dimensione sociale del problema, questo sarà svolto in un ambiente meno pervaso dai Community Smells.

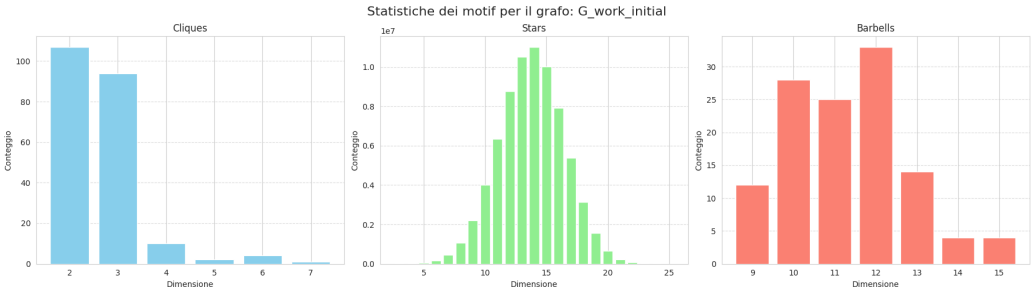

Figure 17: Distribuzione di motivi nel grafo Collaborativo PRIMA dell'ottimizzazione.

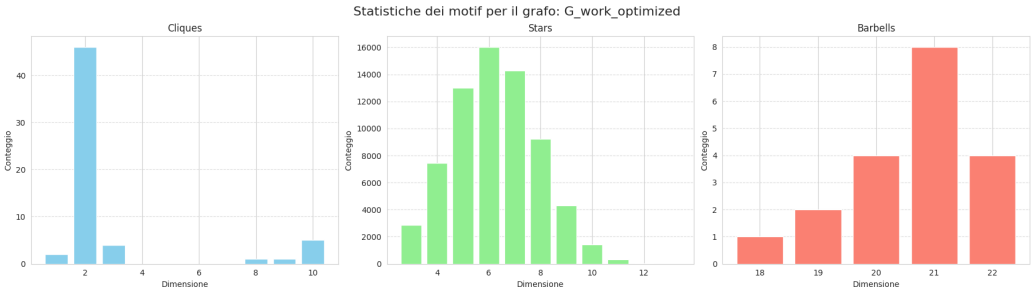

Figure 18: Distribuzione di motivi nel grafo Collaborativo DOPO dell'ottimizzazione.

Test Bottleneck Resolver 1.2 - 134 dipendenti Questo test presenta i miglioramenti introdotti dal Bottleneck Resolver, il csv di partenza è di 134 dipendenti, quindi i grafi avranno 134 nodi.
Di seguito i vincoli scelti per l'ottimizzazione:

bc_threshold_comms: 0.05,

bc_threshold_workprefe: 0.02,

eigen_threshold_social: 0.1,

max_new_edge:30,

max_promoted_number:10,

cost_edge:10.0,

cost_promotion:100.0,

MaxNewEdgesPerBottleneck:10.

Di seguito i parametri di Gurobi per questo test:

TimeLimit: 600,

MIPGap: 0.01,

OutputFlag: 1,

MIPFocus:1,

Heuristics:0.5,

MinNewEdgesIfPromoted:2,

Weight_Comm_Benefit:100.0,

Weight_Node_Quality:10.0,

Weight_Promotion_Value:50.0.

Questo ottimizzatore agisce solo sul grafo comunicativo perchè sono stati è stato possibile riconoscere solo i Bottleneck comunicativi come Community Smells, ed è stato deciso che la modalità migliore per risolverli è quella di aggiungere canali di comunicazione dove mancano. Dopo l'ottimizzazione le metriche sono rimaste sostanzialmente invariate, è stato osservato solo un leggero aumento della closeness. Il comportamento atteso per i motivi è quello della scomparsa delle barbells, ma dal test è stato riscontrato il comportamento opposto, cioè le barbells sono aumentate.

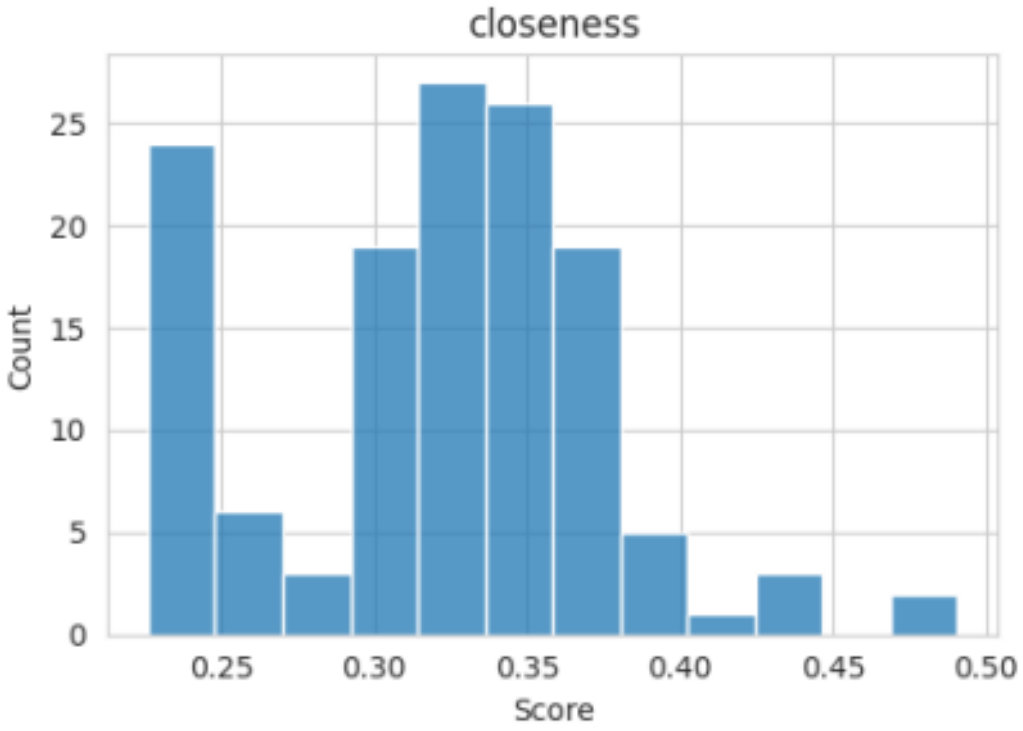

Figure 19: Distribuzione di Closeness del grafo Comunicativo PRIMA dell'ottimizzazione.

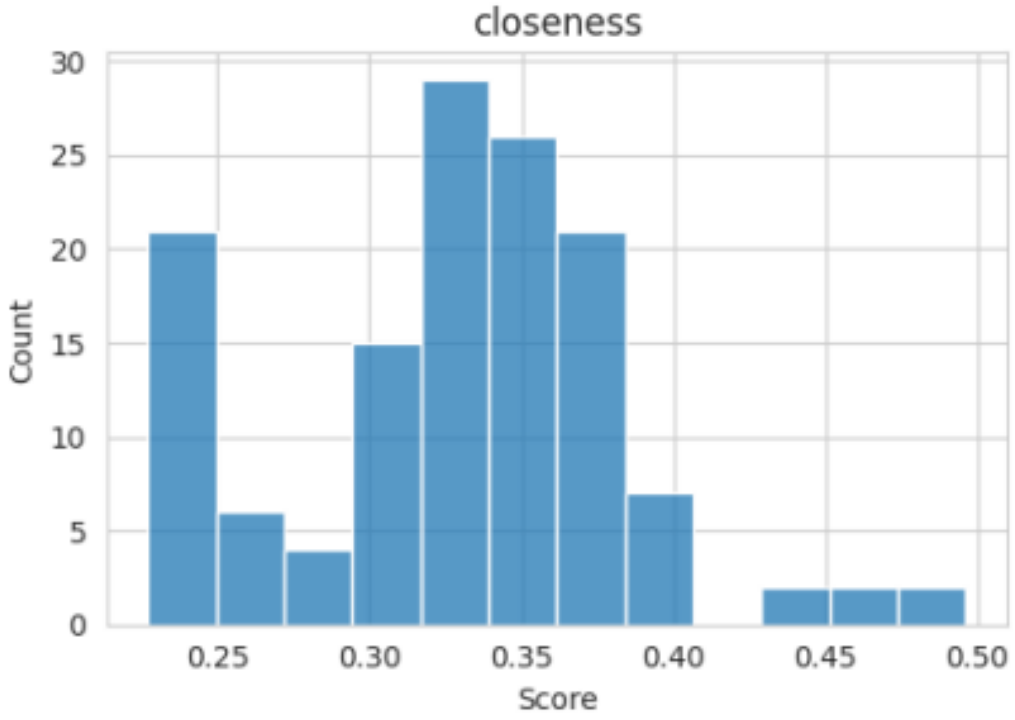

Figure 20: Distribuzione di Closeness del grafo Comunicativo DOPO dell'ottimizzazione.

8 Limiti della soluzione

Il limite riscontrato dalla soluzione implementata è quello che, a valle delle modifiche suggerite da Perfume all'organizzazione dei team aziendali per rimuovere i Silo Organizzativi, il grafo collaborativo risultante risulta completamente connesso, sia per i test effettuati su dataset di venti dipendenti circa, sia per i test effettuati su dataset con più di 100 dipendenti. Questa limitazione potrebbe essere dovuta alla presenza di troppi vincoli cosiddetti *"Hard"*, che prevaricherebbero sui vincoli *"Soft"* andando a trovare sempre la soluzione per cui il

grafo collaborativo è completamente connesso.
Nonostante questa soluzione massimizzi le metriche della rete e, di fatto, faccia scomparire i Silo Organizzativi, non necessita di un risolutore per essere trovata ed è una soluzione poco praticabile nella realtà: ciò vorrebbe dire che, in un progetto di sviluppo software, ogni dipendente dell'azienda lavora su ogni singolo artefatto software, cosa che, anche in una open community, risulta complessa da ottenere e dispendiosa a livello di risorse.
Come ultima nota, gli strumenti a disposizione erano dei semplici notebook, con una potenza di calcolo sicuramente inferiore a quella necessaria per ottenere risultati concreti dall'ottimizzazione di un modello con numerose variabili, quindi, per evitare di danneggiare le macchine, l'esecuzione dell'ottimizzazione è stata limitata a 10 minuti (*TimeLimit: 600*, dove 600 sono i secondi). Sicuramente, con macchine migliori a disposizione e un lasso di tempo più lungo per ottimizzare il modello, il resolver sarebbe in grado di trovare soluzioni ancora migliori di quelle presentate nella sezione dei test.

8.1 Possibile approccio

Uno degli approcci adottati è stato quello di rilassare alcuni vincoli *"hard"* permettendo all'ottimizzatore di riuscire ad avvalersi anche dei vincoli *"Soft"*, che sono stati potenziati in una seconda versione del resolver. A ciò è stato aggiunto un sistema di penalità volto a tenere il risolutore lontano dalla soluzione in cui il grafo risulta completamente connesso.

8.2 Risolutore di Silo Organizzativi 1.2

Il risolutore assegna i dipendenti $i \in E$ a team $t \in T$, e a insiemi di artefatti $k \in A$, massimizzando la coesione sociale e il rispetto delle preferenze, sotto vincoli di dimensione, anzianità e appartenenza.

Variabili decisionali

$$x_{i,t} = \begin{cases} 1 & \text{se il dipendente } i \text{ è assegnato al team } t \\ 0 & \text{altrimenti} \end{cases}$$

$$y_t = \begin{cases} 1 & \text{se il team } t \text{ è attivo} \\ 0 & \text{altrimenti} \end{cases}$$

$$a_{i,k} = \begin{cases} 1 & \text{se } i \text{ lavora sull'artefatto } k \\ 0 & \text{altrimenti} \end{cases}$$

Vincoli principali

Di seguito vengono presentati i vincoli del resolver, a cui è stato aggiunto un sistema di penalità quando ci si allontana troppo dal grafo di partenza. Ciò serve a scoraggiare il resolver a preferire un grafo completamente connesso ad una soluzione ottimale differente.

Ogni dipendente è assegnato esattamente ad un solo team:

$$\sum_{t \in T} x_{i,t} = 1 \quad \forall i \in E$$

La dimensione di ciascun team deve rispettare i limiti minimo e massimo consentiti:

$$\text{minsize} \cdot y_t \;\le\; \sum_{i \in E} x_{i,t} \;\le\; \text{maxsize} \cdot y_t \quad \forall t \in T$$

Il numero totale di team attivi deve rispettare i vincoli globali di quantità:

$$\text{minteams} \;\le\; \sum_{t \in T} y_t \;\le\; \text{maxteams}$$

Ogni team deve avere almeno un dipendente con anzianità superiore alla soglia stabilita:

$$\sum_{i \in E : senior(i) \ge \theta_{\text{senior}}} x_{i,t} \;\ge\; z_t \quad \forall t \in T$$

Un dipendente può lavorare su un artefatto solo se appartiene al team a cui tale artefatto è assegnato:

$$a_{i,k} \;\le\; x_{i,team(k)} \quad \forall i \in E,\; k \in A$$

Funzione obiettivo

La funzione obiettivo massimizza la qualità della suddivisione dei dipendenti nei team:

$$\max \underbrace{\sum_{(i,j)\in F} w_{ij}\left(1 - \sum_{t} x_{i,t}x_{j,t}\right)}_{\text{amicizie}} + \underbrace{\beta \sum_{(i,j)\in W} \sum_{t} x_{i,t}x_{j,t}}_{\text{preferenze di lavoro}} + \underbrace{\gamma \sum_{t\in T} z_t}_{\text{bonus senior}} - \underbrace{\lambda \sum_{i,k} \delta_{i,k}}_{\text{cambiamenti agli artefatti}}$$

In particolare: - il primo termine tiene conto della vicinanza delle coppie amichevoli, - il secondo valorizza le coppie di dipendenti che desiderano lavorare insieme, - il terzo premia la presenza di almeno un senior in ogni team, - l'ultimo penalizza i cambiamenti rispetto alle assegnazioni originali degli artefatti.

8.2.1 Risultati della modifica

Come risultato della modifica della funzione obiettivo e dei vincoli del risolutore, il grafo collaborativo ottimizzato non è più completamente connesso, però non ottiene un miglioramento significativo né nelle metriche misurate, né nelle distribuzioni di motivi riconosciuti nel grafo collaborativo risultante.

9 Sviluppi futuri

La versione di Perfume presentata è solo un prototipo volto a dimostrare che, in linea teorica, l'utilizzo di un risolutore per un modello matematico che rappresenta un'azienda, può ottimizzare la strategia organizzativa dell'azienda che permette una maggiore resistenza ai Community Smells, sfruttando anche l'ingegneria sociale. Visto che il funzionamento di Perfume ha senso soltanto rispettando molte assunzioni, considerando anche che questa è solo una prima formulazione della soluzione, Perfume può sicuramente essere arricchito di molte funzionalità.

Attualmente i risolutori lavorano con grafi orientati (e non) i cui archi hanno dei pesi che possono essere solo positivi, ma una relazione sociale può essere anche negativa, il che implicherebbe la gestione di pesi negativi. Il risolutore in questione potrebbe attivare una funzione che ristrutturerebbe il team, ma soltanto se la quantità di relazioni negative tra i membri del team stesso superassero una determinata soglia. Un'altra proprietà interessante potrebbe essere quella di monitorare la qualità degli artefatti software, in modo da attivare le funzioni di risoluzione di Perfume se la qualità di alcuni di essi dovesse scendere troppo, suggerendo tempestivamente le modifiche che apporterebbero miglioramenti alla qualità del lavoro, riducendo la comparsa dei Community Smells.

Le modifiche suggerite da Perfume avvengono considerando soltanto i dipendenti dell'azienda, ma all'interno di processi di produzione software Perfume potrebbe tranquillamente inserire Agenti I.A. per affiancare gli sviluppatori nella produzione di codice semplice o ricorrente, oppure per potenziare i canali di comunicazione permettendo ai dipendenti un più semplice accesso alle informazioni di interesse, oltre ad evitare l'interruzione del lavoro dei colleghi per una semplice richiesta di informazioni reperibili dalla documentazione.

Infine, si vuole suggerire una prova sul campo di Perfume, che permetterebbe una raccolta di dati adatti come input al software, ma soprattutto si potrebbe fare "fine-tuning", ovvero capire se un suo utilizzo periodico tiene alta la qualità del lavoro in una o più aziende, permettendo un'evoluzione positiva del software. Ciò permetterebbe anche di appurare se l'idea di considerare le relazioni sociali tra i dipendenti nella ricomposizione dei team all'interno di un'azienda è efficace affinché si abbassi la probabilità di occorrenza di un Community Smell.