Corso di Metodi e Modelli per l'Ottimizzazione Combinatoria - Relazione progetto

Universitá degli studi di Padova
Anno Accademico 2015/2016

Studente: Giacomo Quadrio
Matricola: 1061566

Sommario

Il progetto del corso di Metodi e Modelli per l'Ottimizzazione Combinatoria consiste nel risolvere un problema che vede coinvolta un'azienda metalmeccanica che produce pannelli forati per la costruzione di quadri elettrici. La foratura di questi è eseguita attraverso un macchinario a controllo numerico dotato di una punta diamantata che, muovendosi sul pannello secondo una sequenza programmata, produce i fori nelle posizioni desiderate. L'obiettivo è quindi quello di individuare la sequenza di foratura ottimale che minimizzi i tempi di produzione, tenendo conto che il tempo necessario per la foratura è lo stesso e costante per tutti i punti.

1. Modello del problema

Il problema oggetto del progetto puó essere formulato come un problema di ottimizzazione su reti di flusso partendo quindi da un grafo G = (N,A). Scegliendo arbitrariamente un nodo di partenza $0 \in N$ impostiamo ad $|N|$ il flusso uscente da esso in modo tale che venga spinto verso altri nodi. Tale operazione ha peró dei vincoli ovvero ciascun nodo, eccetto l'origine, riceverá una e una sola unitá di flusso, ogni nodo sia visitato una e una sola volta e che il costo del cammino, in termini di pesi c_{ij}, sia minimo.

1.1. Il modello nella Programmazione Lineare Intera

Il problema può essere formalizzato con il seguente modello di programmazione lineare intera. Avremo quindi:

Insiemi

Preprint submitted to Journal Name *2 febbraio 2016*

- **N:** nodi del grafo, rappresentano le posizioni dei fori da realizzare
- **A:** insieme degli archi (i,j) con i e j $\in$ N. Essi rappresentano il tragitto per spostarsi dal nodo i al nodo j

Parametri

- **c_{ij}:** tempo impiegato per spostarsi dal nodo i al nodo j con i e j $\in$ N e l'arco (i,j) $\in$ A.
- **0:** nodo di partenza del cammino $\in$ N.

Variabili decisionali

- **x_{ij}:** unità di flusso trasportate da i a j con i e j $\in$ N e l'arco (i,j) $\in$ A.
- **y_{ij}:** indica l'utilizzo dell'arco (i,j), 1 se viene utilizzato, 0 altrimenti. Avremo che i e j $\in$ N e l'arco (i,j) $\in$ A.

Vincoli

Il modello, a questo punto, prevede un totale di cinque vincoli differenti che possono essere indicati come segue:

1. Il flusso uscente da x_{0j} deve essere massimo, cioè $|N|$
2. Ogni nodo utilizza al massimo una unità di flusso, tranne il nodo di partenza
3. Ogni nodo ha un solo arco in entrata
4. Ogni nodo ha un solo arco in uscita
5. Se vi è un'unità di flusso trasportata da i a j deve di conseguenza esserci un arco che va da i a j

Modello

Inserire formule varie

2. Metaeuristiche scelte per il modello e loro implementazione

Il progetto da svolgere richiesto dal corso di Metodi e Modelli per l'Ottimizzazione Combinatoria prevede l'implementazione del modello di programmazione lineare intera tramite due tecniche ovvero CPLEX ed una o più metaeuristiche a nostra scelta. Una volta fatto ciò si procederà testando i metodi con delle istanze di prova ed i risultati e statistiche confrontati tra di loro per valutarne le prestazioni.

Nello specifico, il problema in esame è un problema di ricerca di vicinato e consiste nel definire una soluzione iniziale e cercare di migliorarla esplorando un intorno di questa soluzione; quindi i metodi utilizzati all'interno del progetto sono due, la Local Search ed il Simulated Annealing, i cui algoritmi sono riportati qui di seguito:

Codice 1: Local Search

```
sol = getInitialSol(random);

while (true){
        vector<int> newSol = findBestN(sol);
        if (evaluate(newSol) >= evaluate(sol)){
                return evaluate(sol);
        }else{
                sol = newSol;
        }
}
```

Come si può vedere, l'algoritmo di Local Search crea innanzitutto una soluzione iniziale da cui partire tramite la funzione getInitialSol, dopodiché esso è stato implementato utilizzando un unico ciclo while che opera finché non viene restituito in output un valore. Questo significa che ciò avverrà unicamente quando, dopo aver calcolato un una nuova soluzione con newSol = findBestN(sol), il suo valore sarà peggiore del valore della soluzione corrente. Se così non è la soluzione corrente viene aggiornata alla soluzione appena calcolata ed il ciclo continua ad operare fino al punto in cui viene trovata una soluzione peggiorativa.

Codice 2: Simulated Annealing

```
sol = getInitialSol(random);
```

```
n_passi = 100000.0 * n / 5;

while (step < n_passi){
        newSol = getNeigh(sol,2,true);
        float de = evaluate(sol) - evaluate(newSol);
        if (de > 0){
                sol = newSol;
        }else{
                temp = 1-(step/n_passi);
                double prob = exp((de)/temp);
                srand(time(NULL)+for_random);
                for_random = for_random + 1;

                if (prob*100 > (rand()%100) ){
                        sol = newSol;
                }
        }
step ++;
}

return(LocalSearch(sol));
```

Per quanto concerne invece l'algoritmo di Simulated Annealing, anch'esso crea innanzitutto una soluzione iniziale da cui partire tramite la funzione getInitialSol, dopodiché troviamo un ciclo while principale che opererà finché step <= n-passi; da notare che il numero di passi inoltre è dinamico così che cresca al crescere del numero di nodi coinvolti nel problema. In cosa consiste quindi questo algoritmo? In sostanza viene calcolata una nuova soluzione attraverso la funzione getNeigh, dopodiché verrà calcolata la differenza tra il valore della soluzione corrente e quello della nuova soluzione. Se il delta ottenuto è maggiore di zero significa che la nuova soluzione è migliorativa e quindi aggiornerò di conseguenza sol, se invece così non è procederemo in maniera differente rispetto alla Local Search. Andremo infatti a calcolare per prima cosa la temperatura di raffreddamento, valore che è coinvolto nel calcolo della probabilità di accettare una mossa peggiorativa. Questa probabilità è calcolata come segue:

$$\text{prob} = \exp(\ \delta/\text{t})$$

dove δ è l'entità del peggioramento de e t è la temperatura temp di raffreddamento. Nel caso la probabilità prob sia maggiore di un numero random calcolato attraverso la funzione srand ciò comporterà appunto l'accettare la mossa peggiorativa, altrimenti essa verrà scartata. Come si modifica però la probabilità p? Essa diminuisce al crescere del peggioramento indotto dalla mossa stessa e cresce al crescere della temperatura t di processo.

Al termine delle operazioni eseguite tramite il ciclo while andremo ad effettuare infine una fase di intensificazione tramite l'operazione di Local Search eseguita sulla soluzione migliore calcolata in precedenza. Il motivo di ciò è che la Local Search è una tecnica relativamente economica in quanto a tempi di esecuzione e permette di raffinare ulteriormente la soluzione trovata.

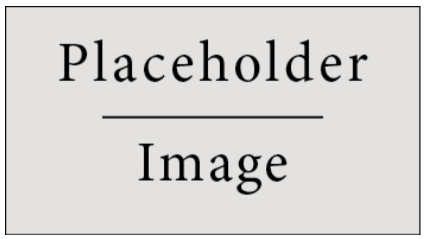

Figura 1: Figure caption

$$e = mc^2 \tag{1}$$

3. Descrizione dei test effettuati

Per verificare le prestazioni degli algoritmi utilizzati nel progetto sono stati condotti diversi test con un numero ben definito di istanze per differenti tipologie di dataset. Nel dettaglio sono stati utilizzati quattro diversi dataset in cui i punti sono disposti come segue:

- **Distribuzione uniforme** ovvero uniformemente disposti all'interno dello spazio
- In **cluster** e disposti in tre raggruppamenti
- In **circoli** e disposti in tre circonferenze
- In **linea** e disposti in tre fasce

Per ciascuna tipologia di dataset sono stati poi selezionati 10 raggruppamenti composti da 4 istanze, ognuno con un numero di nodi che va da 10 a 100 ed in cui la differenza della quantità di elementi tra un gruppo ed il successivo è pari a 10.

I test condotti per i vari algoritmi sono stati effettuati utilizzando, di volta in volta, sempre gli stessi dataset così da poter confrontare meglio i risultati. Ogni istanza di ogni raggruppamento di ogni dataset è stata infine data in pasto agli algoritmi per un totale di 10 volte ed i risultati registrati ritenuti interessanti per la valutazione sono stati quindi i seguenti:

Esecuzione utilizzando CPLEX:

- Tempo medio impiegato per le 10 esecuzioni di ogni istanza
- Valore dell'ottimo

Esecuzione utilizzando la Local Search

- Tempo medio impiegato per le 10 esecuzioni di ogni istanza
- Tempo totale impiegato dalle 10 esecuzioni
- Miglior soluzione ottenuta con la sola Local Search
- Peggior soluzione ottenuta con la sola Local Search
- Media delle soluzioni ottenute con la sola Local Search
- Varianza ottenuna

Esecuzione utilizzando il Simulated Annealing:

- Tempo medio impiegato per le 10 esecuzioni di ogni istanza
- Tempo totale impiegato dalle 10 esecuzioni
- Miglior soluzione ottenuta con il Simulated Annealing
- Peggior soluzione ottenuta con il Simulated Annealing
- Media delle soluzioni ottenute con il Simulated Annealing

- Varianza ottenuna
- Miglior soluzione ottenuta con l'esecuzione della Local Search sui risultati del Simulated Annealing
- Peggior soluzione ottenuta con l'esecuzione della Local Search sui risultati del Simulated Annealing
- Media delle soluzioni ottenute con l'esecuzione della Local Search sui risultati del Simulated Annealing
- Varianza ottenuna

4. Descrizione dei risultati ottenuti