**RICERCA LOCALE**

Con la ricerca sistematica che abbiamo già visto non è facile attaccare problemi complicati e risulta difficile gestire spazi molto grandi di ricerca o infiniti in tali casi si preferisce la ricerca locale. Con la ricerca locale per gestire la complessità dobbiamo sacrificare altri aspetti di efficienza. Molte delle tecniche che si utilizzano sono tecniche che appartengono alla Ricerca Operativa o all’AI e si basano sulle seguenti idee:

1. Passi di miglioramento
2. Passi casuali
3. Ripartenza con assegnazioni differenti

Un algoritmo generale di ricerca locale prende in input un insieme di variabili che caratterizzano il problema, il dominio di appartenenza dei valori delle variabili e un insieme di vincoli che dovranno essere soddisfatti per trovare la soluzione.

Text, timeline

Description automatically generated

Il walk è un ciclo nel quale si effettuano passi casuali che dovrebbero permettere di avvicinarsi alla soluzione, essi consistono in piccoli cambiamenti che permettono di continuare a guardare nello spazio delle soluzioni prossimo alla precedente assegnazione che avevamo. Ogni iterazione fatta dal repeat rappresenta un tentativo(try), nel quale si assegna un valore di partenza alle variabili, questa assegnazione casuale prende il nome di random restart il nome è significativo del risultato che produce questa azione , in fatti il random restart permette di spostarsi da una zona dello spazio si ricerca ad un'altra la quale può essere abbastanza lontana da essere totalmente scorrelata con la precedente assegnazione, in alternativa è possibile usare delle euristiche per migliorare il processo di assegnazione. Il walk che risulta essere un ricerca locale nello spazio delle assegnazioni, esso termina nel momento in cui vengono ad essere soddisfatti i vincoli passati in input, la condizione di stop potrebbe ad ogni modo non essere garantita è l’algoritmo diverge senza incontrare una soluzione. Questo significa che affinché l’algoritmo di ricerca locale sia completo necessita che i criteri di selezione e di stop siano accurati. Gestendo i criteri è possibile avere delle varianti:

RANDOM SAMPLING: ripete indefinitamente il ciclo più esterno andando a provare assegnazioni casuali al fine di trovarne una che soddisfi tutti i vincoli. Lo stop\_walk è sempre true quindi non esegue mai la passeggiata. Questa strategia di non entrare nel walk gli permette di garantire la completezza dell’algoritmo a scapito del tempo richiesto per elaborare soluzione.

RANDOM WALK: in questa variante lo stop\_walk() è sempre a false per cui in questa situazione non si ripetano tentativi di ripartenze casuali andando ad assegnare casualmente dei valori alle variabili in gioco, ma si parte immediatamente con la passeggiata casuale dalla quale uscirà solo quando troverà una soluzione. All’interno del walk seleziona casualmente una delle variabili e prova a darle una assegnazione differente. Questo fa sì che sia più veloce rispetto al random sampling, l’algoritmo risulta anche in questo caso completo.

MASSIMO MIGLIORAMENTO ITERATIVO o ITERATIVE BEST IMPROVEMENT: con questa variante eseguiamo una ricerca locale la quale andrà a selezionare il miglior successore sulla base di una funzione di valutazione ovvero una funzione euristica che fornisca una stima alla conoscenza euristica si unisce la strategia greedy che sceglie il meglio al momento e non sul lungo termine.

A seconda che la funzione euristica sia una funzione che lavori sul costo (quindi bisogno cercare di minimizzarlo) o della qualità (in tal caso andremo a massimizzare tale funzione) otteniamo 2 varianti del iterative best improvement:

1. GREEDY DESCENT 🡪 minimizza il costo
2. GREEDY ASCENT [noto come HILL CLIMBING] 🡪 massimizza la qualità

Nel hill climbing ci sono alcune problematiche che potrebbero bloccare il processo di elaborazione della soluzione e tali condizioni sono:

* Massimi locali
* Creste: una serie di massimi locali molto difficili da gestire e ispezionare.
* Plateau: è un’area piatta del panorama dello spazio degli stati.

Questi algoritmi cercano OTTIMI LOCALI ove nel caso del greedy descent si parla di minimi locali, mentre nel caso del greedy ascent parliamo di massimi locali. La funzione quindi deve cercare di minimizzare in ambedue i casi il numero dei conflitti.

L’algoritmo potrebbe risultare incompleto nel caso in cui trovo 2 ottimi locali che fanno in modo di intrappolare l’algoritmo da un ottimo locale all’altro. Questo caso porta ad incompletezza e per sfuggire a queste situazioni ci serve ripartire in modo casuale da un'altra zona dello spazio di ricerca per poi proseguire nella ricerca della soluzione.

Il successo del hill climbing dipende molto dalla forma del panorama dello spazio degli stati: se ci sono pochi massimi locali e plateau, la variante con il riavvio casuale troverà la soluzione molto velocemente.

ALGORITMI STOCASTICI

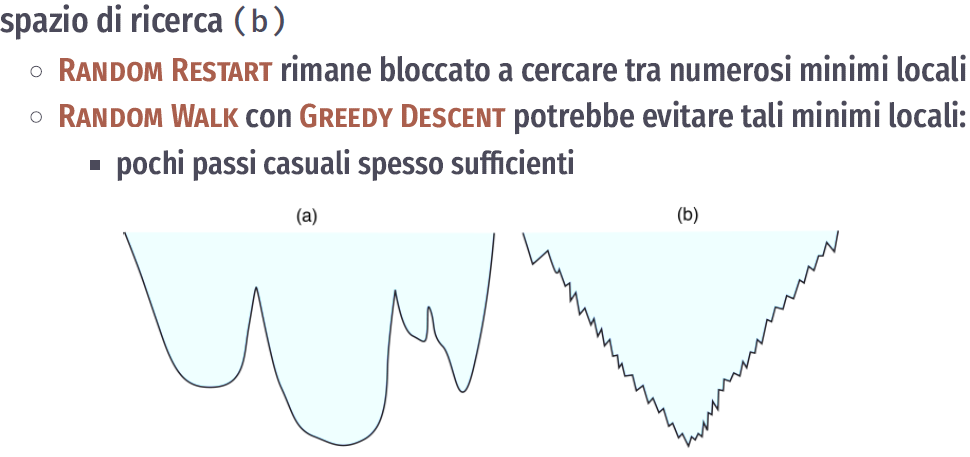
L’obiettivo è quello di evitare i minimi locali che non siano globali e per farlo si usa la casualità andando ad unire al massimo miglioramento iterativo delle mosse random che danno vita ad una ricerca locale stocastica.

Questa strategia porta a :

1. RANDOM RESTART
2. RANDOM WALK 🡪 eseguito nel greedy descent e nel greedy ascent



Il random restart permette di evadere dagli avvallamenti profondi che portano minimi locali del grafico come viene mostrato in figura (a).



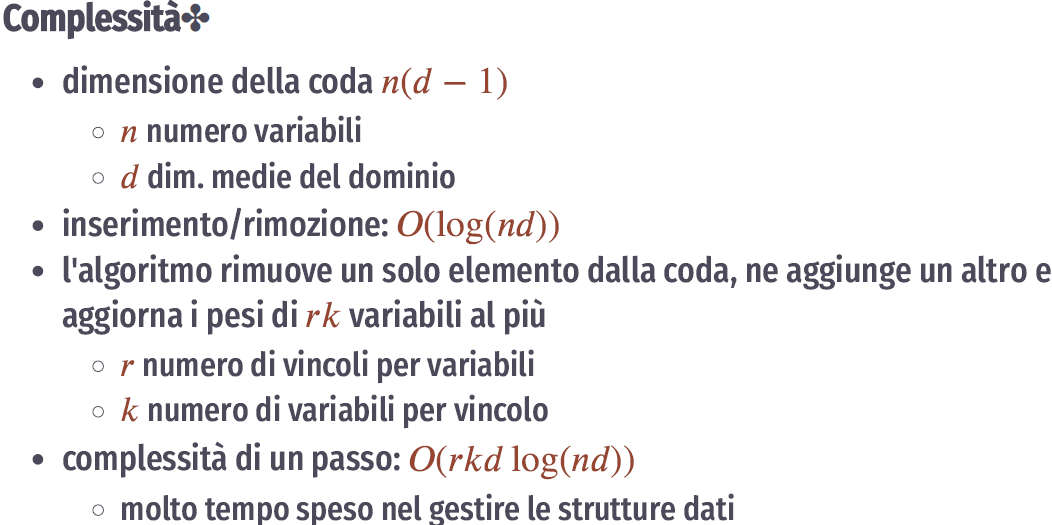
In questo altro spazio di ricerca risulta più idoneo e performante il random walk il quale cercando nello spazio di ricerca limitrofo sarà capace di superare una serie di minimi locali posti gli uni vicini agli altri, situazione nella quale il random restart avrebbe grossi problemi visto che rimarrebbe incastrato spostandosi da un minimo locale ad un altro senza mai riuscire a puntare verso il minimo globale, come si evince nella figura (b).

VARIANTI DELLA RICERCA LOCALE

In base alla funzione che sceglierà il successore da testare avremo differenti varianti, per cui la funzione euristica definirà differenti modi di agire da parte di un algoritmo di ricerca locale.

TABU SEARCH: si usa una struttura dati ausiliare per ricordare le ultime modifiche effettuate (supponiamo negli ultimi t passi svolti) in modo che tali variabili che sono state cambiante in questi passi non le andrò a cambiare. La lista delle ultime t variabili modificate prende il nome di tenure e viene aggiornata ad ogni passo e permette anche di evitare cicli. L’idea del tabu search e in particolare dell’uso della tenure list è quella di mantenere memoria sulla direzione della ricerca questo porta l’algoritmo a capire la direzione dalla quale proveniva nell’esplorazione dello spazio di ricerca e a muoversi nella ricezione corretta senza tornare indietro sui suoi passi. Questo meccanismo serve al tabu search per affrontare i plateau riuscendo a superarli.

PASSO DI MASSIMO MIGLIORAMENTO: si selezionano coppie variabile-valore che portano al miglioramento della funzione di valutazione. Se ci sono più coppie che portano ad un miglioramento se ne sceglie una casuale. Effettuando la scelta casuale si effettua un approccio di forza bruta a tentativi; invece, utilizzando una coda con priorità è possibile scegliere le coppie variabile-valore più promettenti e tale stima è data da un peso che permetterà di gestirle nella coda prioritaria.



SCELTA A DUE FASI: simile al precedente, qui l’idea è quella di scegliere prima la variabile e in seguito scegliere quale valore assegnarle. Per effettuare la prima scelta sulle variabili si utilizza una coda con priorità ove il peso e dato dal numero di conflitti in cui tali variabili sono coinvolte ad ogni passo si prenderà la variabile che partecipa a più conflitti e su questa variabile si cambia il valore assegnato cercando di minimizzare l’assegnamento a valori che rispettino i vincoli nei quali è implicata la variabile selezionata. È computazionalmente più pesante, ma permette di convergere prima alla soluzione.

Graphical user interface, text, application

Description automatically generated

ANY CONFLICT: l’idea è quella di distinguere le variabili conflittuali ovvero che sono coinvolte nei vincoli da quelle che non lo sono e ad ogni iterata si selezionano casualmente dall’insieme delle variabili conflittuali una per andarle ad assegnare o un valore casuale oppure un valore che tenda a massimizzare i vincoli imposti andandoli a rispettare. Per il criterio di scelta è possibile prima scegliere il conflitto che siamo interessati a minimizzare e in seguito una delle variabili coinvolte in tale conflitto.

Text

Description automatically generated

Più una variabile è coinvolta in più conflitti e più avrò possibilità di selezionarla.

SIMULATED ANNEALING: questo algoritmo prende il nome da un processo che viene normalmente eseguito nelle industrie che lavorano i metalli. L’idea che c’è dietro la metafora con la metallurgia e la termodinamica è che a temperature più alte un metallo è malleabile facilmente al proprio volere e quindi al cambio di direzione, al contrario a basse temperature il metallo tende a irrigidirsi e difficilmente effettuerà dei cambiamenti. Il simulated annealing risulta essere una combinazione di hill climbing e algoritmi di esplorazione casuale come il random restart per mettere assieme gli aspetti migliori di ognuno di questi algoritmi. Si basa su di un algoritmo di ricerca locale stocastico il quale inizialmente adotterà una maggiore casualità nella scelta dei conflitti selezionati per assegnare valori a tali variabili, successivamente più avanti nella ricerca dell’algoritmo quando avrà trovato una zona di maggiore interesse si focalizzerà su di essa diminuendo la casualità della scelta delle variabili conflittuali e quindi incentrando la ricerca nella zona limitrofa con l’obiettivo di trovare l’ottimo globale. Questo algoritmo alle alte temperature si comporta come un random walk che cerca di trovare la regione dello spazio delle soluzione ove si potrà trovare l’ottimo globale. Successivamente a temperature basse si comporta come un greedy descent che punta dritto a trovare l’ottimo locale sperando che al passo precedente si sia scelta la zona ove sia presente l’ottimo globale.

Text

Description automatically generated

[A = è l’assegnazione fatta sulle variabili]

DISTRIBUZIONE DI GIBS-BOLTZMAN

A picture containing text

Description automatically generated

La tecnica del random restart può essere utilizzata per migliorare un algoritmo casuale debole o, meglio, un algoritmo che riesce ad avere buona efficienza nel trovare la soluzione, ma fallisce poiché si concentra solo su zone ristrette dello spazio di ricerca, per cui rieseguendolo iterativamente e ad ogni ri-esecuzione partendo da una zona differente grazie al random restart è possibile migliorare di molto le prestazione in termini di completezza ed efficacia dell’algoritmo casuale debole. In alcuni casi è possibile pensare ad un partial restart.

ALGORITMI BASATI SU POPOLAZIONI

In questo paragrafo andiamo ad affrontare dei metodi avanzati di ricerca locale che tendono a lavorare in parallelo per ottimizzare i tempi nel rilevamento di una soluzione

Questi metodi vanno a gestire direttamente popolazioni ovvero insiemi di individui ove gli individui altro non sono che le assegnazioni fatte. Quindi anziché lavorare su una variabile per volta e sulla migliore assegnazione che è possibile dare si portano avanti con più variabili assieme, le quali costituiscono una popolazione o fascio e si lavora su di esse.

BEAM SEARCH: questo algoritmo va a lavorare su un fascio di individui in particolare sui migliori k individui presenti, quello che cerca di fare è portare contemporaneamente avanti più ricerche locali per volta Ad ogni iterata si portano avanti i migliori k successori. Quello che banalmente fa questo algoritmo è iterare l’algoritmo di ricerca locale invece che su di una sola assegnazione su di un insieme di assegnazioni per volta. Infatti, se k fosse = 1 allora il beam search si trasforma in un greedy search. Risulta utile quando la memoria a disposizione è limitata. Per migliorare le prestazioni del beam search è possibile usare una funzione di valutazione euristica che permette di portare avanti le migliori k assegnazioni in base al valore associato dalla funzione di valutazione. Una particolare variante è il beam search stocastico dove i k individui vengono ad essere selezionati in modo casuale favorendo quelli che hanno una valutazione migliore. Stiamo dicendo che la principale differenza tra il beam search e lo stochastic beam search sta nel come avviene l’assegnazione, mentre nel local search le mutazioni sulle variabili avvengono governate da una funzione euristica che ne minimizza il costo ovvero va a ridurre i conflitti, nello stochastic beam search le mutazioni sono aleatorie il che significa che c’è la possibilità di accettare assegnazioni peggiorative con il fine di esplorare diverse zone dello spazio di ricerca delle soluzioni. Di seguito riportiamo la funzione di scelta dello stochastic beam search che si basa sulla distribuzione di Gibbs proprio come avveniva per il simulated annealing.

Graphical user interface, text, application

Description automatically generated

H() ovvero la funzione di valutazione riflette l’adattamento dell’individuo ovvero il fitness, valuta le migliori o peggiori probabilità dell’individuo di andare avanti e portare ad una soluzione. Il fitness è un indicatore della capacità di sopravvivenza di quell’individuo ergo significa che è una misura della bontà dell’assegnazione effettuata.

In una ricerca local beam, l’informazione utile viene passata dall’uno all’altro thread di ricerca parallela. Per cui tale algoritmo permette di avvisare gli altri theard e di congentrarli in uno spazio di ricerca ove sono presenti individui con maggiore speranza di portare a soluzioni.

ALGORITMI GENETICI: questo algoritmo va a portare avanti nel calcolo i migliori k individui ai fini riproduttivi. Ovvero ad ogni iterata rimescola le n coppie di individui dove n = k/2 e k è il numero degli individui di partenza e quindi delle assegnazioni di partenza al fine di migliorarle nelle generazioni successive ove le generazioni sono iterate dell’algoritmo. In questi algoritmi si utilizza la tecnica del crossover per incrociare e rimescolare le popolazioni parentali che danno vita alla popolazione dei figli.

Il crossover può essere:

* Crossover uniforme: vi è pari probabilità di copiare una variabile che provenga da un genitore oppure dall’altro. Qui in questo caso per ogni variabile dell’assegnazione che andrà a costituire il figlio la variabile in questione ha equa probabilità di essere ereditata da uno dei due genitori
* One-point crossover: in questo caso si assegna un indice, il quale permette di capire fino a che punto copiare una parte di variabili che sarà presa da un genitore mentre la restante parte sarà presa dall’altro.

Gli algoritmi evolutivi possono essere visti come una variante della ricerca beam stocastica nei quali si applica la metafora biologica ove esiste una popolazione e tra gli individui più favorevoli alla sopravvivenza si viene a generare la prole attraverso un processo di ricombinazione, in questo caso la ricombinazione è data dal cross-over al quale è possibile aggiungere piccole variazioni casuali, ma solo per alcune variabili. Gli algoritmi evolutivi possono variare in base a:

* Dimensioni della popolazione
* Rappresentazione di ciascun individuo
* Numero di genitori da i quali si forma la prole sono usati 1 singolo genitore, 2 coppia di genitori, ma è possibile usare un numero p>2 che rappresenta l’insieme parentale.
* Processo di selezione e punteggio di fitness
* Procedura ricombinate
* Tasso di mutazione
* Elitismo una tecnica che permette di mantenere immutabile l’adattabilità che per tale ragione non diminuirà nella prole.
* Abbattimento [CULLING] in cui tutti gli individui al di sotto di una certa soglia di fitness sono scartati.

Nel meccanismo del Cross-over si usa il concetto di schema per la formazione di nuovi individui. Le rappresentazioni conformi allo schema sono dette istanze e sono possibili individui della prole che si andrà a generare.

OTTIMIZZAZZIONE

Il problema di ottimizzazione per essere definito deve avere un criterio di ottimalità con il quale andare a selezionare la soluzione migliore da fornire. Questi criteri di ottimalità vengono visti come vincoli soft che impongono delle condizioni di preferibilità sulle soluzioni trovate.

