Proposta Tesi, tirocinio Argo Software

Angelo Battaglia

May 27, 2021

1 Introduzione

Citando Luenberger and Ye [1], un problema di ottimizzazione riguarda l'allocazione delle risorse e l'analisi della complessità di un problema. L'ottimizzazione si suddivide in tre sottoinsiemi: Linear Programming, Unconstrained Problems e Constrained Problems. In seguito, utilizzerò questi nomi sia in Italiano che in Inglese. Il successo delle tecniche di ottimizzazione della programmazione lineare risiedono soprattutto nella formulazione del problema, piuttosto che nel metodo utilizzato per risolverlo. Un problema con un grande numero di vincoli, infatti, presenta usualmente delle caratteristiche di linerità.

1.1 Linear Programming

Siano date i entità, alle quali associamo

$$w_i$$

pesi. Un problema base che riguarda la programmazione lineare è:

$$massimizzare \quad w_1x_1 + w_2x_2$$

$$soggetto\ al\ vincolo \quad x_1 + x_2 \leq B$$

$$con \quad x_1 \geq 0,\ x_2 \geq 0$$

1.2 Unconstrained Problems

Problemi senza vincoli, ovvero unconstrained problems, sono quelli che considerano tutto il fascio di decisioni possibili. L'assenza di vincoli amplia la portata del problema, lo generalizza, aumentandone i gradi di libertà. Inoltre, aggiungere dei vincoli a questa classe di problemi è un'operazione che non presenza, di solito, particolari criticità. A livello teorico, i problemi senza vincoli assumono un particolare valore teorico, che successivamente va a concretizzarsi nella quotidianità dei problemi con vincoli.

1.3 Constrained Problems

I problemi ai vincoli, detti Constrained Problems, trovano più utilità nelle applicazioni. I vincoli vengono imposti nei problemi complessi per semplificarli, e per discretizzarli. Ogni vincolo riduce la portata del problema. Un problema generale che riguarda la programmazione matematica è:

$$minimizzare \quad f(\mathbf{x})$$

$$soggetto\ ai\ vincoli \quad h_i(\mathbf{x}) = 0,\ i = 1, 2, ...,\ m$$

$$g_j(\mathbf{x}) = 0,\quad j = 1, 2, ...,\ p$$

$$con\ x \in S$$

In questa definizione, $\mathbf{x}$ è un vettore n-dimensionale nelle incognite x_i, della forma $\mathbf{x} = (x_1, x_2, ..., x_n)$, ed

$$f\ ,\ h_i,\ i = 1, 2, ...,\ m,\ g_j,\ j = 1, 2, ...,\ p$$

sono delle funzioni a valori reali delle variabili $x_1,\ x_2,\ ...,\ x_n$. Una misura ovvia della complessità di un problema è la sua dimensione. La dimensione di un problema è misurata nei termini del numero di incognite e di vincoli. Come ci si può aspettare, i problemi trovano una più facile soluzione quando la potenza dei calcolatori aumenta e la teoria matematica viene raffinata. Un problema di larga scala viene ad avere migliaglia o milioni di variabili e vincoli. Un possibile strada da seguire, in termini di software, è *Gecode, Generic Contraint Development Environment* [2].

1.4 Algoritmi Iterativi e convergenza

La caratteristica più importante di un super-computer è la sua abilità di computare operazioni ripetitive in maniera efficiente. Per questa ragione, molti molti algoritmi di ottimizzazione, per loro natura, sono iterativi. Tipicamente si parte da un vettore iniziale x_0 per poi ottenerne uno, tramite un algoritmo, x_1. Il processo è ripetuto finché una soluzione migliore x_2 viene trovata. Continuando finché Una sequenza di punti sempre migliori è trovata $(x_0,\ x_1, ...,\ x_k, ...)$ che si avvicini alla soluzione x^*. Per i problemi di programazione lineare risolti dal metodo del simplesso, la sequenza generata è di lunghezza finita, e raggiunge la soluzione dopo un finito numero di passi, anche se non inizialmente specificato. Per la programmazione non lineare, la sequenza non raggiunge il punto x^* soluzione, ma converge verso esso, finché il processo termina ad un punto sufficientemente vicino al punto della soluzione.

1.5 Convergence-rate theory

Abbiamo due aspetti della convergence-rate theory. Il primo è generalmente conosciuto come analisi della complessità, e concerne la velocità dell'algoritmo di convergenza, distinguendo tra tempo polinomiale e tempo non polinomiale. Il secondo aspetto provvede l'analisi di quanto il metodo velocemente converge nei suoi passi finali, e può provvedere un paragone dei vari metodi.

2 Tabu Search

As *Tabu Search* [3] states, Tabu S

2.1 Simulated Annealing

Come cita l'articolo di Bertsimas and Tsitsiklis [4], il Simulated Annealing è un algoritmo di ottimizzazione che usa diversi strumenti matematici, come

3 Algoritmi Genetici

Bibliografia e crediti

[1] David G. Luenberger and Yinyu Ye. *Linear and Nonlinear Programming*. Springer, 2015.

[2] *Gecode, Generic Contraint Development Environment*. Gecode. 2021. URL: `https://www.gecode.org/index.html`.

[3] *Tabu Search*. Wikipedia. 2021. URL: `https://en.wikipedia.org/wiki/Tabu_search`.

[4] Dimitris Bertsimas and John Tsitsiklis. "Simulated Annealing". In: *Statistical Science* 8.5 (1993), pp. 10–15.