Algoritmi di ricerca con subottimalità vincolata

Francesco Percassi, Alfonso E. Gerevini

Università degli studi di Brescia

October 6, 2020

Classi di algoritmi di ricerca sub-ottima

- Gli algorimi sub-ottimi sono classificati in base alle garanzie che vengono richieste sulla qualità delle soluzioni prodotte. Possiamo distinguere:
  - **Any solution**: non viene richiesta alcuna garanzia sulla qualità dei piani. Esempio: Greedy Best First Search (GBFS).
  - **Bounded Suboptimal Search** (**BSS**): trovare una soluzione tale che $cost \leq B * cost_{opt}$, con: $B \geq 1$ e $cost_{opt}$ è il costo della soluzione ottima
    - Una soluzione è B-ammissibile se soddisfa la disuguaglianza $cost \leq B * cost_{opt}$
  - **Bounded Cost Search** (**BCS**): trovare una soluzione tale che $cost \leq C$
    - Esempio applicativo: *Expedia*, l'utente richiede un volo per una specifica destinazione che abbia un costo limitato ad un budget desiderato

BSS - FOCAL Search

- Gli algoritmi proposti per garantire la *B-ammissibilità* delle soluzioni sono riconducibili ad una classe di algoritmi denominata **Focal Search** e fanno uso di euristiche **ammissibili**;

- **Euristica ammissibile**: non sovrastima mai il costo necessario per raggiungere il goal:

$$\forall n \; h(n) \leq h^*(n)$$

- Viene mantenuta separatamente una lista FOCAL $\subseteq$ OPEN;

- FOCAL $= \{n \in OPEN \mid g(n) + h(n) = f(n) \leq B \cdot f_{min}\}$ con:

$$f_{min} = \underset{n \in \text{OPEN}}{\text{argmin}} \; f(n)$$

- l'estrazione dei nodi è limitata ai nodi contenuti in FOCAL.

BSS - FOCAL Search

Algorithm 1: Focal Search: main procedure

```
1  focal-search(start state S)
2      OPEN ← {S};
3      FOCAL ← {S};
4      while FOCAL ≠ ∅ do
5          best ← ChooseNode(FOCAL)
6          Remove best from FOCAL and OPEN
7          if best is a goal then return best;
8          if f_min increased then FixFocal();
9          for n ∈ neighbors(best) do
10             Add n to OPEN
11             if f(n) ≤ B × f_min then add n to FOCAL;
12         end
13     end
14 end
```

- Progredendo la ricerca il valore f_{min} potrebbe crescere e parte dei nodi della OPEN vengono inclusi in FOCAL mediante `fixFocal()`
- Gli algoritmi di ricerca si differenziano dalla definizione della funzione `ChooseNode(FOCAL)`, se questa sceglie il minimo: A^*_ϵ [Pearl, 1982]

BSS - Weighted A* Search [Pohl, 1970]

- Algoritmo di ricerca BSS che esplora prioritariamente i nodi aventi la funzione $f_W(n)$ minima:

$$f_W(n) = g(n) + W \cdot h(n)$$

dove:

  - $g(n)$ costo minimo del cammino conosciuto per raggiungere n dallo stato iniziale;
  - $h(n)$ euristica - ammissibile - che stima il costo per raggiungere il goal;
  - $W \geq 1$; se $W = 1$, $WA^* = A^*$, se $W \to \infty$ allora $WA^* \to GBFS$;

- WA^* è un caso speciale di Focal Search, la soluzione trovata è W-ammissibile [Gilon et al, 2016].

BSS - Explicit Estimation Search (EES) [Thayer, 2011]

- **Separazione dei ruoli**: uso di euristiche **inamissibili** per guidare la ricerca e **ammissibili** per garantire la *B-ammissibilità*; superamento di A^*_ϵ

- Combinazione della stima **cost-to-go** a **distance-to-go** per orientare la ricerca verso soluzioni *B-ammissibili* più veloci da raggiungere;

- euristiche impiegate: $h(n)$ (*cost-to-go* ammissibile), $\hat{h}(n)$ (*cost-to-go* non ammissibile), $\hat{d}(n)$ (*distance-to-go* non ammissibile)

- Per ogni nodo n definiamo le seguenti quantità:
  - $f(n) = g(n) + h(n)$ [COSTO]
  - $\hat{f}(n) = g(n) + \hat{h}(n)$ [COSTO]
  - $\hat{d}(n)$ [DISTANZA]

BSS - EES (scelta del nodo)

- Durante la ricerca viene tenuta traccia dei seguenti nodi notevoli:
  - $best_f = \underset{n \in \text{OPEN}}{\text{argmin}}\ f(n)$
  - $best_{\hat{f}} = \underset{n \in \text{OPEN}}{\text{argmin}}\ \hat{f}(n)$
  - $best_{\hat{d}} = \underset{n \in \text{OPEN} \wedge \hat{f}(n) \leq B \cdot \hat{f}(best_{\hat{f}})}{\text{argmin}}\ \hat{d}(n)$

Ad ogni espansione il nodo viene scelto sulla base delle seguenti regole:

1. **if** $\hat{f}(best_{\hat{d}}) \leq B \cdot f(best_f)$ **then** scegli $best_{\hat{d}}$
2. **else if** $\hat{f}(best_{\hat{f}}) \leq B \cdot f(best_f)$ **then** scegli $best_{\hat{f}}$
3. **else** scegli $best_f$

BCS - Potential Search [Stern, 2011]

- **Potential Search** algoritmo nativo per il framework BCS;
- **Obiettivo:** Espandere prioritariamente i nodi seconda la probabilità che questi siano parte di un piano avente costo *pari o inferiore* a C;
- Espansione nodi basata sulla nozione di potenziale calcolata come:

$$u(n) = \frac{C - g(n)}{h(n)}$$

- Intuitivamente: a parità di euristica scegliamo il nodo avente $g(n)$ minimo, a parità di costo $g(n)$ scegliamo il nodo avente euristica minima

Dynamic PS - from BCS to BSS [Gilon et al, 2016]

- I framework BSS e BCS sono strutturalmente simili;
- Un algoritmo BCS è **ragionevole** quando ha una struttura *best-first* (dotato di una lista OPEN e di una regola di espansione) e ogni nodo n avente $f(n) \geq C$ viene scartato mediante il pruning.
- ogni algoritmo per BCS *ragionevole* può essere migrare per il contesto BSS (e viceversa);
- **Dynamic Potential Search**: ridefinizione della funzione potenziale

$$u_C(n) = \frac{C - g(n)}{h(n)} \Rightarrow u_B(n) = \frac{B \times f_{min} - g(n)}{h(n)}$$

Anytime Search

- **Algoritmi anytime**: producono iterativamente delle soluzioni di qualità crescente

- Tipologia di algoritmi anytime:
  - *Continued Search*: ricerca best-first che termina quando la lista OPEN è esaurita;
  - *Repairing Search*: ad ogni soluzione trovata l'algoritmo rimodula i suoi parametri per orientare la ricerca verso soluzioni di maggiore qualità;
  - *Restarting Search*: ogniqualvolta viene trovata una soluzione la ricerca viene riavviata mantenendo tutti gli sforzi precedentemente calcolati.

Anytime Search - Algoritmi Noti

- **ANA*, Anytime Non-parametric A*** (*continued search*): Adattamento anytime dell'algoritmo PS nel quale viene espanso il nodo che massimizza la quantità:

$$u(n) = \frac{G - g(n)}{h(n)}$$

dove G è il costo dell'ultima soluzione trovata (*incumbent solution*).

- **RwA*** usato in **LAMA** [Richter et al, 2010] (*restarting search*): ricerca greedy *distance-to-go* seguita da una successione di ricerche WA* *cost-to-go* (basate da una coppia di euristiche inammissibili) con peso decrescente $W_{list} = [5, 3, 2, 1]$.

Anytime EES [Thayer, 2012]

- EES: algoritmo BSS che garantisce la B-ammissibilità della soluzione prodotta;
- **Idea**: eseguire una ricerca EES con dei bound calcolati automaticamente dopo ogni soluzione trovata (evitando uno schedule di parametri statici come RwA* (LAMA);
- **Obiettivo**: soddisfare una nozione ad hoc di *performance ideale*

Performance ideale di un algoritmo anytime

Performance Ideale

Dato un algoritmo χ, indichiamo con π_i^χ l'i-esima soluzione trovata da χ al tempo $t(\pi_i^\chi)$. Un algoritmo anytime realizza un comportamento *ideale* quando minimizza il tempo $t(\pi_{i+1}^\chi) - t(\pi_i^\chi)$.

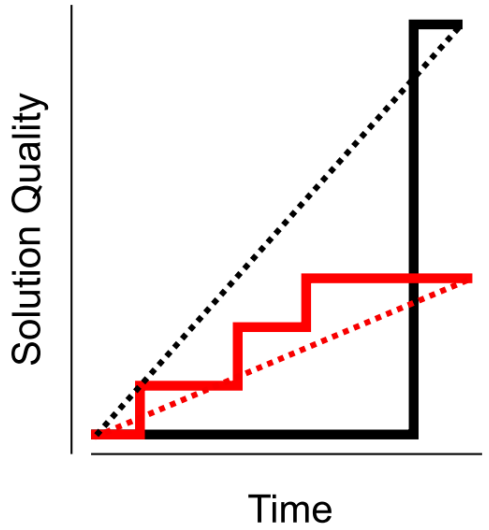

Figure 2: Maximizing $\frac{\Delta q}{\Delta t}$ is not ideal.

Figure: Figura tratta da [Thayer, 2012]

Anytime EES - Pseudo codice

Il bound dell'i-esima ricerca EES viene calcolato dinamicamente secondo la seguente formula:

$$w = \frac{cost}{f(best_f)} = \frac{cost}{LB}$$

AEES(root)

1. $open \leftarrow \{root\}, cost \leftarrow \infty, w \leftarrow \infty$
2. **while** $open \neq \{\}$
3. let $n = selectNode(open, w)$ in
4. **if** $f(n) \geq cost$ **then** continue
5. **else if** $goal_p(n)$ **then** $onGoal(n, w, cost, open)$
6. **else** $expand(n, open, cost)$
7. $open \leftarrow open - \{n\}$
8. **for each** child c of n
9. **if** $f(c) < cost$ **then** $open \leftarrow open \cup \{c\}$

onGoal(n, w, cost, open)

1. **if** $g(n) < cost$
2. **then** let $best_f = \text{argmin}_{n \in open} f(n)$ in
3. $cost \leftarrow g(n)$
4. $w \leftarrow \frac{cost}{f(best_f)}$

selectNode(open, w)

1. **if** $\widehat{f}(best_{\widehat{d}}) \leq w \cdot f(best_f)$ **then** $best_{\widehat{d}}$
2. **else if** $\widehat{f}(best_{\widehat{f}}) \leq w \cdot f(best_f)$ **then** $best_{\widehat{f}}$
3. **else** $best_f$

Un approccio per gestire stime (soft-bound) (i)

Scenario

Supponiamo di avere una stima del costo di una soluzione per un problema di ricerca (pianificazione).

Tale stima:

- può essere calcolata da un esperto di dominio oppure mediante un predittore;
- pertanto può essere significativamente inaccurata;
- l'inaccuratezza non ci permette di poter utilizzare direttamente gli algoritmi pensati per BSS e BCS.

Challenge: *come possiamo sfruttare questa stima per migliorare le performance di un algoritmo di ricerca?*

Un approccio per gestire stime (soft-bound) (ii)

- Ingredienti: $g(n)$, $h_{search}(n)$ non ammissibile, $h_{\delta}(n)$ ammissibile e C costo stimato;

- Schema di ricerca adottato wA^*:

$$f_W(n) = g(n) + W * h_{bound}(n)$$

con $h_{bound}(n) = F(g(n), h_{search}(n), h_{\delta}(n), C)$;

Un approccio per gestire stime (soft-bound) (iii)

- L'euristica sensibile ai costi è stata progettata per aumentare la velocità di convergenza dell'algoritmo:

$$h_{bound}(n) = h_{search}(n) * \left(\frac{C}{g(n) + h_{\delta}(n)} \right)$$

- **Fattore moltiplicativo** permette di penalizzare e premiare i nodi
- La ricerca viene accelerata verso quei nodi che appaiono essere prossimi a C:
  - tutti i nodi tali per cui $g(n) + h_{\delta}(n) \ll C$ sono penalizzati
  - tutti i nodi che giacciono oltre $g(n) + h_{\delta}(n) \geq C$ sono scontati

Come trattare stime inaccurate?

- Se $C \rightarrow C_{opt}$ rischiamo di concentrare la ricerca in uno spazio dove è molto difficile trovare una soluzione

- **Soluzione proposta**: introduzione della metrica $penalty_{rate}$:

$$penalty_{rate} = \frac{\{\#exp.nodes \mid g(n) + h_\delta(n) > C\}}{\#exp.nodes}$$

$$h_{bound}(n) = h_{search}(n) * \left(\frac{C}{g(n) + h_\delta(n)} \right)^{(1-p_{rate})}$$

- L'uso del p_{rate} "spegne" gradualmente l'effetto di C sull'euristica in favore di h_{search}

Conclusioni

- Abbiamo formalizzato la nozione di algoritmi subottimi vincolati:
  - (BSS) Focal Search: A^*_ϵ, WA^*, Explicit Estimated Search (EES)
  - (BCS) Potential Search (PT)
- BSS e BCS perseguono lo stesso obiettivo e gli algoritmi per un setting sono facilmente convertibili per l'altro sotto certe condizioni
- Algoritmi anytime: ANA*, RwA*, LAMA, Anytime EES (AEES)
- Presentazione di un algoritmo in grado di gestire stime senza garanzie (soft-bound)

Bibliografia

Pohl, Ira

Heuristic search viewed as path finding in a graph.
Artificial intelligence, 1:193–204 1970.

Pearl, Judea and Kim, Jin H

Studies in semi-admissible heuristics.
IEEE transactions on pattern analysis and machine intelligence, 4:392–399, 1982.

Thayer, Jordan Tyler and Ruml, Wheeler

Bounded suboptimal search: A direct approach using inadmissible estimates.
Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI 2011), 674–679 2011.

Thayer, Jordan Tyler and Ruml, Wheeler

Potential Search: A Bounded-Cost Search Algorithm.
Proceedings of the Twenty-First International Conference on Automated Planning and Scheduling, (ICAPS 2011), 234–241 2011.

Thayer, Jordan Tyler and Ruml, Wheeler

Dynamic Potential Search - A New Bounded Suboptimal Search.
Proceedings of the Ninth Annual Symposium on Combinatorial Search, (SOCS 2016), 36–44 2016.

Richter, Silvia and Westphal, Matthias

The LAMA planner: Guiding cost-based anytime planning with landmarks.
Journal of Artificial Intelligence Research (JAIR), 39:127–177.

Richter, Silvia and Westphal, Matthias

Better Parameter-Free Anytime Search by Minimizing Time Between Solutions.
Proceedings of the Fifth Annual Symposium on Combinatorial Search, (SOCS 2012), 120–128 2012.