Algoritmi di Ricerca con Sub-Ottimalità Vincolata

Seminario per il corso di IA 2023 - 2024

Alfonso E. Gerevini[1], **Francesco Percassi**[2],

[1] Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Brescia, Italia
[2] School of Computing and Engineering, University of Huddersfield, UK

Table of Contents

Preliminari

Preliminari [i]

Problema di Search

Un **problema di search** è una tupla $\langle S, s_{init}, succ(s), c(s, s'), G \rangle$ dove:

- S è un insieme di stati;
- $s_{init} \in S$ è lo stato iniziale;
- $succ(s)$ è la funzione che restituisce i **successori** di s, i.e., $succ(s) \subseteq S$;
- $c(s, s')$ è la funzione che associa alla transizione da s a s' un **costo**;
- $G \subseteq S$ è l'insieme dei goal.

Obiettivo

Trovare un **path** da s_{init} ad uno stato in G. Il **costo** di questo path è uguale al costo di tutte le transizioni che ne fanno parte.

Preliminari [ii]

Nodo

Un **nodo** n rappresenta uno stato più il path necessario per raggiungerlo a partire da s_{init}. Indichiamo con $g(n)$ il **costo** di questo path.

Euristica cost-to-go, Euristica Perfetta & Euristica distance-to-go

Dato un nodo n, denotiamo con:

- $h(n)$, la stima del costo per raggiungere G (**cost-to-go**);
- $d(n)$, la stima del numero di transizioni per raggiungere G (**distance-to-go**);
- $h^\star(n)$, **l'euristica perfetta**.

Euristica Ammissibile

Una euristica si dice **ammissibile** se, per ogni nodo n, $h(n) \leq h^*(n)$. *Non ammissibile* in caso contrario.

Best-First Search, Greedy Best First Seach (GBFS) & $A^\star$ [Hart et al., 1968]

- Strategia di ricerca **informata** per risolvere un problema di search.

Function `Best-First-Search`(**start state** s_{init}):
  OPEN $\leftarrow \{s_{init}\}$;
  while *OPEN* $\neq \emptyset$ **do**
    $best \leftarrow$ `chooseNode`(OPEN);
    OPEN = OPEN $\setminus \{best\}$;
    if $best \in G$ **then**
      return $best$;
    end
    OPEN $\leftarrow$ OPEN $\cup\, succ(n)$
  end

- se `chooseNode` usa $h(n)$: GBFS
- se usa $f(n) = g(n) + h(n)$: $A^\star$;
  - se $h(n)$ **ammissibile**, $A^\star$ produce soluzione **ottima**.

Classificazione Problemi Sub-Ottimi

Classifichiamo un algoritmo **sub-ottimo** in funzione delle garanzie che fornisce sulle soluzioni prodotte:

- **Any Solution**: non viene imposto alcun requisito, e.g., GBFS;
- **Sub-ottimalità vincolata**:
  - **Bounded Cost Search** (**BCS**): vincoliamo *esplicitamente* il costo delle soluzioni;
  - **Bounded Sub-Optimal Search** (**BSS**): vincoliamo il costo in modo *parametrico*.

Bounded Cost Search (BCS) [Stern et al., 2011]

- BCS vincola **esplicitamente** il costo delle soluzioni valide.

Problema di Search BCS

- Definizione: $\langle S, s_{init}, succ(s), c(s, s'), G, C \rangle$ dove C è un **costo**;
- un path è una soluzione **valida** se e soltanto se $c(path) \leq C$;
- se non esiste una soluzione tale per cui $c(path) \leq C$, il problema è **irrisolvibile**.

Bounded Sub-Optimal Search (BSS)

- BSS vincola in modo **parametrico** le soluzioni considerate valide, senza conoscere a priori un valore indicativo C.

Problema di Search BSS

- Definizione: $\langle S, s_{init}, succ(s), c(s, s'), G, w \rangle$ dove w è un **parametro** che controlla la sub-ottimalità;
- un path è una soluzione **valida** se e soltanto se $c(path) \leq w \cdot C_{opt}$, dove $w \geq 1$ e C_{opt} è il costo del path ottimo; tale soluzione è detta w-**ammissibile**;

- **Intuizione**: se $w = 2$ ammettiamo come valide tutte quelle soluzioni il cui costo non eccede il doppio del costo della soluzione ottima.

Perchè la sub-ottimalità vincolata è importante?

- Molti problemi di search sono **computazionalmente impossibili** da risolvere in modo ottimo;
  - e.g., planning;
- in molti ambiti non siamo interessati a trovare una soluzione ottima;
  - ci "accontentiamo" di una soluzione sub-ottima purché rispetti certi requisiti;
- la sub-ottimalità vincolata pone le medesime sfide della search ottima ma è **computazionalmente più scalabile**.

Algoritmi BSS

Focal Search [i]

- Tutti gli algoritmi per BSS sono varianti della **Focal Search**;
  - Per garantire w-ammissibilità, ingrediente fondamentale: $h(n)$ ammissibile.

- **Caratteristiche chiave**:
  - Oltre alla frontiera OPEN, abbiamo la FOCAL$\subseteq$ OPEN:

$$\text{FOCAL} = \{n \in \text{OPEN} \mid g(n) + h(n) \leq w \cdot f_{min}\}$$

$$f_{min} = \min_{n \in \text{OPEN}} f(n)$$

  - l'estrazione dei nodi è limitata alla FOCAL list.

- **Importante**: f_{min} è un **lower bound** per il costo ottimo, dunque:

$$g(n) + h(n) \leq w \cdot f_{min} \leq w \cdot C_{opt}$$

Focal Search [ii]

- Al progredire della ricerca, f_{min} cresce e parte dei nodi transitano da OPEN a FOCAL (funzione `Fix-Focal()`);
- Gli algoritmi Focal Search si differenziano in base alla funzione `ChooseNode`;
- se viene scelto il nodo con $f(n)$ minimo, algoritmo: $A^{\star}_{\epsilon}$ [Pearl and Kim, 1982];

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
```

Figure: Algoritmo Focal Search. Fonte: [Gilon et al., 2016].

Weighted $A^\star$ Search [Pohl, 1970]

- $wA^\star$ **generalizza** $A^\star$ cambiando la funzione impiegata per decidere quale nodo espandere prioritariamente; dato $w \geq 1$:

$$f(n) = g(n) + w \cdot h(n)$$

  - le soluzioni prodotte sono w-ammissibili;
  - se $w = 1$, $wA^\star = A^\star$;
  - se $w \to \infty$, $wA^\star \to$ GBFS (i nodi prioritarizzati unicamente secondo $h(n)$)

- $wA^\star$ è un caso speciale di Focal Search (che non fa uso esplicito di FOCAL).

Debolezza di $A^\star_\epsilon$ (e $wA^\star$) per BSS

- Entrambi questi algoritmi usano una sola euristica con un duplice ruolo:
  1. guidare la ricerca verso il goal;
  2. garantire la w-ammissibilità.
- **Explicit Estimation Search** (EES) [Thayer and Ruml, 2011]
  - **Intuizione**: disaccopiare, da un punto di vista euristico, ① e ②;
  - combinare **cost-to-go** (w-ammissibilità) con **distance-to-go** (goal).

Explicit Estimation Search (EES) [Thayer and Ruml, 2011]

- Ingredienti per EES:
  - $h(n)$, euristica ammissibile *cost-to-go*;
  - $\hat{h}(n)$, euristica non ammissibile *cost-to-go*;
  - $\hat{d}(n)$, euristica non ammissibile *distance-to-go*.

- Per selezionare quale nodo espandere vengono usate le funzioni:

costo : $f(n) = g(n) + h(n)$;
costo : $\hat{f}(n) = g(n) + \hat{d}(n)$;
distanza : $\hat{d}(n)$;

Explicit Estimation Search (EES) - Selezione Nodo

- Durante la search viene tenuta traccia dei seguenti nodi notevoli:
  - $best_f = \underset{n \in \text{OPEN}}{\text{argmin}}\, f(n)$
  - $best_{\hat{f}} = \underset{n \in \text{OPEN}}{\text{argmin}}\, \hat{f}(n)$
  - $best_{\hat{d}} = \underset{n \in \text{OPEN} \wedge \hat{f}(n) \leq w \cdot \hat{f}(best_{\hat{f}})}{\text{argmin}}\, \hat{d}(n)$
- Ad ogni espansione viene scelto un nodo tra $\{best_f, best_{\hat{f}}, best_{\hat{d}}\}$ secondo:
  1. **se** $\hat{f}(best_{\hat{d}}) \leq w \cdot f(best_f)$ **allora** espandi $best_{\hat{d}}$;
  2. **se** $\hat{f}(best_{\hat{f}}) \leq w \cdot f(best_f)$ **allora** espandi $best_{\hat{f}}$;
  3. **altrimenti** espandi $best_f$

EES vs $wA^\star$

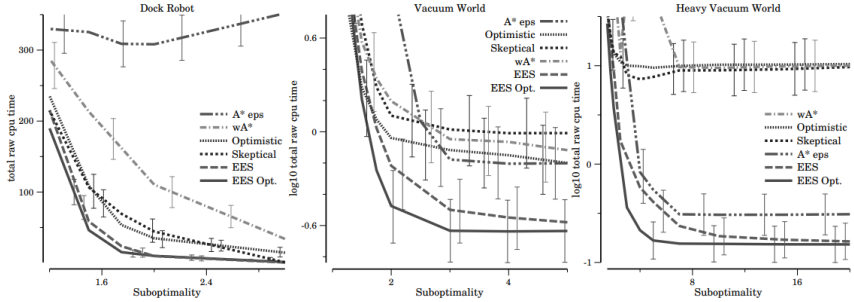


Figure: CPU Runtime (asse y) per EES & $wA^\star$ al variare della sub-ottimalità w (asse x). Fonte: [Thayer and Ruml, 2011].

Algoritmi BCS & Migrazione

Potential Search (PS) [Stern et al., 2011]

- **Memo**: una soluzione è valida nel framework BCS se $c(path) \leq C$;
- **Intuizione**: espandiamo prioritariamente i nodi che **massimizzano la probabilità** che questi siano parte di un path avente costo $\leq$ C;
- **Funzione potenziale**:

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

  - a parità di $h(n)$ scegliamo il nodo avente $g(n)$ minimo;
  - a parità di $g(n)$, scegliamo il nodo avente $h(n)$ minimo.

- Scartiamo (**potiamo**) tutti i nodi tali per cui $g(n) + h(n) > C$

Migrazione da BCS a BSS (e viceversa)

- BCS e BSS sono **strutturalmente** simili;
- **Differenza**:
  - BCS, vincolo **statico** ($g(n) + f(n) \leq C$);
  - BSS, vincolo **dinamico** ($g(n) + f(n) \leq w \cdot f_{min}$);
- Possiamo **migrare** ogni algoritmo per BCS a BSS purché:
  - abbia una struttura best-first;
  - poti tutti i nodi che eccedono C.

Dynamic Potential Search (DPS) [Gilon et al., 2016]

- Adattiamo la funzione potenziale $u(n)$ di PS:

$$u(n) = \frac{C - g(n)}{h(n)} \Rightarrow u(n) = \frac{w \cdot f_{min} - g(n)}{h(n)}$$

- **Proprietà teorica**: similmente a $wA^\star$, DPS produce soluzioni w-ammissibili senza esplicitare FOCAL;

- **Problema**: gestione dei valori euristici in OPEN che cambiano dinamicamente al variare f_{min}.

Da BSS a BCS - Bounded-Cost Explicit Estimation Search (BEES) [Thayer et al., 2012b]

- Similmente, un algoritmo per BSS può essere adattato al contesto BCS;
- Il nodo avente **priorità massima** in EES, i.e., $best_{\hat{d}}$, viene ridefinito come:

$$best_{\hat{d}} = \underset{n \in \text{OPEN} \wedge \hat{f}(n) \leq w \cdot \hat{f}(best_{\hat{f}})}{\text{argmin}} \hat{d}(n) \Rightarrow best_{\hat{d}_C} = \underset{n \in \text{OPEN} \wedge \hat{f}(n) \leq C}{\text{argmin}} \hat{d}(n)$$

- **Selezione Nodi**:

  se esiste nodo $n \in$ OPEN t.c. $\hat{f}(n) \leq C$ **allora** scegli $best_{\hat{d}_C}$
  altrimenti scegli $best_f$

Algoritmi Anytime

Algoritmi Anytime

- **Contesto**: produrre una soluzione **approssimata** entro un **walltime**;
- **Strategia Anytime**:
  - produrre delle soluzioni di qualità **crescente**;
  - maggiore walltime, maggiore qualità delle soluzioni prodotte;
  - *anytime*, in quanto possono essere interrotti in qualunque momento.

Strategie per Algoritmi Anytime

- **Continued Search**:
  si prosegue nella ricerca fino a che non si **esaurisce** OPEN;

- **Repairing Search**:
  per ogni soluzione trovata si **rimodulano** i parametri della ricerca per orientarla verso soluzioni di maggiore qualità;

- **Restarting Search**:
  per ogni soluzione trovata si **riavvia** la ricerca.

Repairing Search - Anytime Repairing $A^\star$ (ARA*) [Likhachev et al., 2003]

- Uso di $h(n)$ ammissibile;
- Uso di $wA^\star$ con w dinamico;
- **Riparazioni** ogniqualvolta viene provata la w-ammissibilità di una soluzione:
  - w viene decrementato (convergenza verso qualità migliore);
  - i valori euristici in OPEN vengono ricalcolati secondo il nuovo w.
- **Debolezza** (scoperta a posteriori): *low h-bias*.

Low h-bias in ARA*

Low h-bias

- ARA* preserva OPEN per ogni *fase* di search;
- ogniqualvolta viene trovata una soluzione, permangono in OPEN molti nodi prossimi al goal s_g con valutazioni **euristiche molto basse**;
- bias *esacerbato* dall'aggiornamento di OPEN mediante il decremento di w.

Conseguenze su OPEN

- poca diversificazione = scarsa esplorazione;
- espansione di stati nell'intorno di s_g.

Restarting Anytime $wA^\star$ - R$wA^\star$ [Richter et al., 2010]

- Basato sull'idea, *controintuitiva*, di **riavviare** la ricerca quando viene trovata una soluzione;
- low h-bias emendato da azzeramento OPEN;
- Sequenza di episodi di ricerca $wA^\star$ con peso decrescente secondo $\mathcal{W} = [5, 3, 2, 1.5, 1]$;
- R$wA^\star$ è stato implementation con successo nel pianificatore LAMA.

Restarting vs Repairing - Risultati Sperimentali

- La congettura sul low h-bias è dimostrata sperimentalmente.

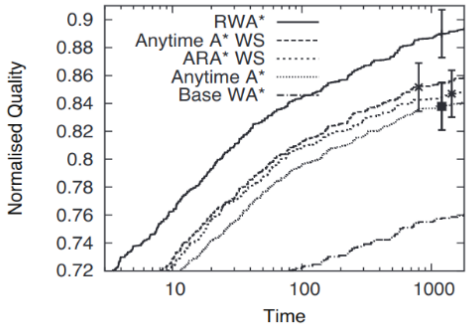


Figure: Qualità soluzioni trovate (asse y) over time (asse x) ARA* e R$wA^\star$. Fonte: [Richter et al., 2010].

Algoritmi Anytime *Principled*

- ARA* e R$wA^\star$ *presuppongono* una conoscenza a priori:
  - Sequenza di pesi w impiegata;
- per ARA*, w_0 settato arbitrariamente e poi decrementato del 2% per ogni fase di search;
- per R$wA^\star$, $\mathcal{W} = [5, 3, 2, 1.5, 1]$;
- **Soluzione**: sfruttare algoritmi BSS/BCS per ottenere degli algoritmi Anytime **metodici**.

Algoritmi Anytime *Principled* - Anytime Non Parametric $A^\star$ (ANA*) [van den Berg et al., 2011]

- Adattamento Potential Search al contesto Anytime;
- **Privo di parametri**;
- Dato l'i-esimo di ricerca episodio, viene espanso il nodo che minimizza la funzione:

$$u(n) = \frac{C_{i-1} - g(n)}{h(n)}$$

  dove C_{i-1} è il costo della soluzione trovata all'episodio $i-1$ (*incumbent solution*);
  - C_0 inizializzato con un valore arbitrariamente grande.

Algoritmi Anytime *Principled* - Anytime EES (AEES) [Thayer et al., 2012a]

- **Memo**: EES è un algoritmo potente per BSS che garantisce w-ammissibilità.
- **Intuizione**: usare EES come mattoncino in contesto anytime
  - Prima iterazione settiamo $w = \infty$, dunque GBFS;
  - Ricalcoliamo w in base al costo dell'ultima soluzione trovata.
- **Vantaggio**: schedule **dinamico** dei parametri $\mathcal{W}$, laddove prima era **statico**.
- **Obiettivo**: Ottenere una **performance ideale** di algoritmo Anytime.

Anytime EES [ii] - Nozione di Performance Ideale

Performance Ideale

Dato un algoritmo χ, denotiamo con π_i^χ la i-esima soluzione trovata dall'algoritmo χ al tempo $t(\pi_i^\chi)$. Un algoritmo Anytime realizza una **performance** ideale quando **minimizza** $t(\pi_{i+1}^\chi) - t(\pi_i^\chi)$

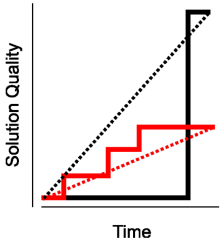


Figure 2: Maximizing $\frac{\Delta q}{\Delta t}$ is not ideal.

Figure: Fonte: [Thayer et al., 2012a].

Anytime EES [iii]

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** $best_f$

Figure: Fonte: [Thayer et al., 2012a]

- In AEES, per ogni soluzione trovata, w viene rimodulato secondo la formula:

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

Ricerca con Soft-Bound

Soft Bound - Contesto

Contesto

Supponiamo di disporre di una **stima (potenzialmente inaccurata)** per un problema di pianificazione. Tale stima può provenire da:

- un esperto del dominio che si intende risolvere;
- generata automaticamente da un **predittore**;

Inoltre:

- può essere significativamente **inaccurata**;
- l'inaccuratezza impedisce di mutuare (direttamente) gli algoritmi BSS/BCS.

Sfida

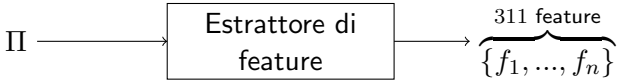
È possibile migliorare le perfomance di un algoritmo di search sfruttando questa stima?

Soft Bound - Predire una Stima [i]

- Dato un problema di planning Π, è possibile estrarre delle **feature** rilevanti;
- Queste feature rappresentano **aspetti strutturali** di Π (*impronta digitale*);
- **Esempi**:
  - numero di variabili, azioni, azioni rimosse, numero di oggetti, etc.
  - landmarks, composizione dei landarmarks, etc.

Soft Bound - Predire una Stima [ii]

- Dato un problema di planning Π, estraiamo **311 feature** prodotte da diverse sorgenti:
  - preprocessing di FastDownard [Helmert, 2006];
  - preprocessing di LPG [Gerevini et al., 2003];
  - codifiche di Π, i.e., PDDL, SAS, SAT;



- Costruiamo un dataset composto da coppie $\langle \overbrace{\{f_1, ..., f_n\}}^{\Pi}, C \rangle$ dove C è il costo di una soluzione per Π e **alleniamo un modello** per predire C.

- L'uso delle feature ci permette di ottenere un predittore **domain-independent**.

Soft Bound - $h^{discount}(\cdot)$ [i]

- Definizione di una euristica $h^{discount}(\cdot)$ **sensibile** alla stima C
- **Ingredienti**:
  - $g(n)$, $h^{search}(n)$ non ammissibile, $h^{\delta}(n)$ ammissibile, C costo stimato;
- Schema di ricerca adottato $wA^{\star}$:

$$f(n) = g(n) + w \cdot h^{discount}(g(n), h^{search}(n), h^{\delta}(n), C)$$

Soft Bound - $h^{discount}(\cdot)$ [ii] [Percassi et al., 2020]

- $h^{discount}$ è progettata per accelerare la convergenza della search verso una soluzione di costo $\geq C$:

$$h^{discount}(\cdot) = h^{search}(n) \cdot \frac{C}{g(n) + h^{\delta}(n)};$$

- h^{search} viene modulata mediante il **fattore moltiplicativo** dimodoché:
  - siano **penalizzati** i nodi tali che $g(n) + h^{\delta}(n) \ll C$;
  - siano **promossi** i nodi tali che $g(n) + h^{\delta}(n) \gg C$.

Soft Bound - Gestione Stime Inaccurate

- Se $C \to C^{opt}$, viene scontata una porzione dello spazio di ricerca molto estesa;
- Se $C \leq C^{opt}$, degenerazione in una **ricerca a costo uniforme**

- **Soluzione proposta**:

$$h^{\textit{discount-pr}}(\cdot) = h^{search}(n) \cdot \left(\frac{C}{g(n) + h^{\delta}(n)}\right)^{1-p_{rate}}$$

$$p_{rate} = \frac{|\{n \in \textsc{ExpNodes} \mid g(n) + h^{\delta}(n) > C\}|}{|\textsc{ExpNodes}|}$$

- $0 \leq p_{rate} \leq 1$ misura la % di nodi espansi il cui lower-bound supera C;
- p_{rate} spegne gradualmente l'effetto di C su h^{search}.

Soft Bound - Risultati Sperimentali

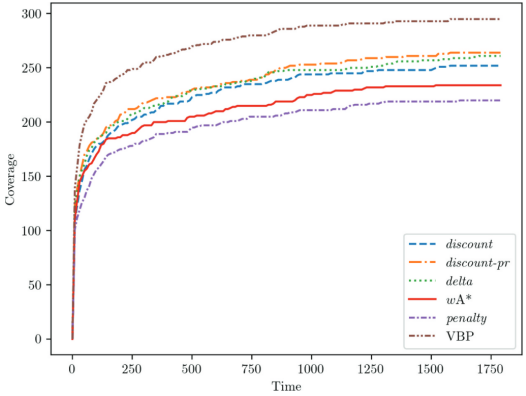


Figure: Coverage (asse y) over Time (asse x) per $wA^\star$ e ricerca condizionata dalla stima ($h^{\textit{discount}}$, $h^{\textit{discount-pr}}$). Fonte: [Percassi et al., 2021].

Conclusioni

Conclusioni

- Formalizzazione di due paradigmi di ricerca sub-ottima vincolata:
  - BSS: Focal-Search, $A^\star_\epsilon$, $wA^\star$ & EES;
  - BCS: PS;
- BSS & BCS perseguono lo stesso obiettivo:
  - migrazione BCS $\rightarrow$ BSS: PS $\rightarrow$ DPS;
  - migrazione BSS $\rightarrow$ BCS: EES $\rightarrow$ BEES;
- Algoritmi Anytime:
  - parametrici: ARA* & R$wA^\star$,
  - non-parametrici: ANA* & AEES;
- Euristiche sensibili ad una stima (soft bound):
  - $h^{\textit{discount}}$ & $h^{\textit{discount-pr}}$;
- **Bonus**: Risultati aggiornati e compendio BSS [Fickert et al., 2022].

References I


Fickert, M., Gu, T., and Ruml, W. (2022).
New Results in Bounded-Suboptimal Search.
In *Proc. of AAAI*, volume 36, pages 10166–10173.


Gerevini, A., Saetti, A., and Serina, I. (2003).
Planning Through Stochastic Local Search and Temporal Action Graphs in LPG.
Journal of Artificial Intelligence Research, 20:239–290.


Gilon, D., Felner, A., and Stern, R. (2016).
Dynamic Potential Search - A New Bounded Suboptimal Search.
In *Proc. of SoCS*, pages 36–44.

References II

Hart, P. E., Nilsson, N. J., and Raphael, B. (1968).
A Formal Basis for the Heuristic Determination of Minimum Cost Paths.
IEEE Trans. Syst. Sci. Cybern., 4(2):100–107.

Helmert, M. (2006).
The Fast Downward Planning System.
Journal of Artificial Intelligence Research, 26:191–246.

Likhachev, M., Gordon, G. J., and Thrun, S. (2003).
ARA*: Anytime A* with provable bounds on sub-optimality.
Advances in Neural Information Processing Systems, 16.

Pearl, J. and Kim, J. H. (1982).
Studies in Semi-Admissible Heuristics.
IEEE Trans. Pattern Anal. Mach. Intell., 4(4):392–399.

References III

Percassi, F., Gerevini, A. E., Scala, E., Serina, I., and Vallati, M. (2020). Generating and Exploiting Cost Predictions in Heuristic State-Space Planning. In *Proc. of ICAPS*, volume 30, pages 569–573.

Percassi, F., Gerevini, A. E., Scala, E., Serina, I., and Vallati, M. (2021). Improving Domain-Independent Heuristic State-Space Planning via Plan Cost Predictions. *J. Exp. Theor. Artif. Intell.*, 35(6):849–875.

Pohl, I. (1970). Heuristic Search Viewed as Path Finding in a Graph. *Artif. Intell.*, 1(3):193–204.

References IV

Richter, S., Thayer, J. T., and Ruml, W. (2010).
The Joy of Forgetting: Faster Anytime Search via Restarting.
In *Proc. of ICAPS*, pages 137–144.

Stern, R. T., Puzis, R., and Felner, A. (2011).
Potential Search: A Bounded-Cost Search Algorithm.
In *Proc. of ICAPS*, pages 234–241.

Thayer, J., Benton, J., and Helmert, M. (2012a).
Better Parameter-Free Anytime Search by Minimizing Time Between Solutions.
In *Proc. of SoCS*, volume 3, pages 120–128.

References V

Thayer, J., Stern, R., Felner, A., and Ruml, W. (2012b).
Faster Bounded-Cost Search Using Inadmissible Estimates.
In *Proc. of ICAPS*, volume 22, pages 270–278.

Thayer, J. T. and Ruml, W. (2011).
Bounded Suboptimal Search: A Direct Approach Using Inadmissible Estimates.
In *Proc. of IJCAI*, pages 674–679.

van den Berg, J., Shah, R., Huang, A., and Goldberg, K. Y. (2011).
Anytime Nonparametric A*.
In *Proc. of AAAI*, pages 105–111.

Grazie per la vostra attenzione.