New Results About Sub-Admissibility for General Families of Heuristic Search Algorithms

Henri Farreny

Institut de Recherche en Informatique de Toulouse (IRIT),
118 Route de Narbonne, 31 062 Toulouse cedex, France. Email : Henri.Farreny@irit.fr

Abstract : We propose a formal generalization for various works dealing with Heuristic Search in state graphs. This generalization focuses on the properties of the evaluation functions, on the characteristics of the state graphs, on the notion of path length, on the procedures that control the choices of the node expansions, on the rules that govern the updating operations. Consequently, we present new theorems about the sub-admissibility. These theorems widely extend the analogous results previously published concerning Nilsson's A* algorithms and diverse successors due to Pohl (HPA), Harris (particular A), Martelli (B), Pearl and Kim (A^*_ε), Ghallab and Allard (A_ε), Bagchi and Mahanti (C), Pearl (BF*), Mero (B′), Korf (IDA*), Mahanti and Ray (D), Dechter and Pearl (A**). They provide a theoretical support for using diverse kinds of Heuristic Search algorithms in enlarged contexts.

1 Position of this Work

According to Simon [28], Search and Reasoning are two complementary ways to view Problem Solving : « A logic taken by itself may be viewed as a non deterministic algorithm for finding all of the consequences of a set of premises… Then the logic must be supplemented by some kind of control structure. A search strategy must be superimposed upon it… In this sense, reasoning processes are a subset of search processes ; the subset that uses rules of inference as the sole operators ». Generally, Search algorithms are guided by heuristics ; commonly, the heuristics use numeric values and functions in order to evaluate and to compare different perpectives. So, Heuristic Search is a possible approach for representing some kinds of knowledge and for managing some kinds of reasoning.

Heuristic Search algorithms are studied from about thirty years. [29] lists one thousand papers concerning this topic, of which several hundreds refer to Heuristic Search in *state graphs*. What is our contribution here ? We do not describe another particular algorithm or a new experimentation. But we propose an unifying and generalizing point of view and we explain how the constraints usually respected by the Heuristic Search algorithms in state graphs, in order to assure the admissibility or the sub-admissibility, can be widely relaxed. More precisely, we present original theorems that can be applied to obtain sub-admissibility formulas when working with more general state graphs, or with more general evaluation functions, or with more general path lengths or with more general algorithmic mechanisms ; for instance, the arc costs and the heuristic estimates may be not exclusively positive, the arc costs of a path may be aggregated by other operations than the addition, the heuristic estimate of any node may be a variable rather than a constant. In particular, these theorems can be applied to rediscover and *extend* the results about admissibility or sub-admissibility concerning Nilsson's A* [15, 20, 21] and many of its successors.

2 Admissibility : Previous Results, Ways for Extensions

First we recall the definitions of Nilsson's A and A* and we report under which precise conditions it has been established that A* terminates by discovering a minimal solution. Then, we recall precise results about the admissibility or sub-admissibility of various descendants of A or A* ; but we do not recall neither the complete definitions nor the other interesting properties of these algorithms.

Finally, we discern several attractive ways to relax the definitions and running conditions of Heuristic Search algorithms.

2.1 Algorithms A

The algorithm described in Fig. 1 is widely known as *algorithm A* [21]. It searchs a goal node by progressively expliciting the state graph from the start node « s ». The state graph is supposed to be *son-finite*, that is to say : the number of sons of any node is finite. A *cost* denoted c(m,n) is associated to each arc (m,n) from a node « m » to a node « n ». The notation A* [20] is considered further.

```
1   procedure A-of-Nilsson
2      open ← {s} ; closed ← {} ; g_memo(s) ← 0 ; f_memo(s) ← h(s)
3      until open = {} repeat
4      |  m ← best-firt-extraction
5      |  if m ∈ T then edit-reverse-path stop endif
6      |  open ← open – {m} ; closed ← closed + {m}
7      |  for each n son of m repeat
8      |  |  g_neo ← g_memo(m) + c(m,n)
9      |  |  if n ∈ open and g_neo < g_memo(n) then update-father&g_memo&f_memo endif
10     |  |  if n ∈ closed and g_neo < g_memo(n) then  update-father&g_memo&f_memo
11     |  |                                             open ← open + {n}
12     |  |                                             closed ← closed – {n} endif
13     |  |  if n ∉ closed and n ∉ open then  update-father&g_memo&f_memo
14     |  |                                   open ← open + {n} endif
15     |  endrepeat
16     endrepeat
17  procedure edit-reverse-path
18     until m = s repeat write m ; m ← father(m) endrepeat ; write "s"
19  procedure update-father&g_memo&f_memo
20     father(n) ← m ; g_memo(n) ← g_neo ; f_memo(n) ← g_memo(n) + h(n)
21  procedure best-first-extraction
22     return node n* of open such as f_memo(n*) = min_{q ∈ open} f_memo(q)
```

Fig. 1. Algorithm A of Nilsson.

The set of the goal nodes is denoted by « T » (line 5) ; if a goal is discovered, the algorithm writes the reversed node list of a path from the start node to the found goal (lines 17, 18) and terminates (stop in line 5). Possibly, the algorithm terminates without discovering any goal node (test in line 3), even if such a node exists in the considered state graph. Possibly, it does not terminate. The execution of the algorithm is constituted by node *extractions* (lines 4 to 6) followed by the *expansions* of the extracted nodes (lines 7 to 15). At the beginning of each execution of the loop 3-16, « open » is the set of the nodes that are available for the next step of the search ; « closed » is the set of the nodes previously extracted and expanded that are not available now. The algorithm uses an *evaluation function* « f » to direct the choice of the node that must be extracted from open and then expanded. We denote « $f_x(n)$ » the later value of the evaluation function at the node « n » when the extraction « x » is about to be executed ; the index « x » is aimed to recall that the same node « n » may be successively evaluated in different ways, according to the successive *extractions* (lines 4 to 6). The algorithm systematically extracts from « open » (lines 4 and 21-22) one of the nodes that own the minimal evaluation ; this mechanism may be interpreted like this : the more the evaluation of a node is small, the more we hope that this node leads a path to a goal. The sons of the extracted node are evaluated (for the first time or afresh) ; any new son is put in « open » ; if a son is already in « closed » and if its present evaluation is less than its previous evaluation then the node is got out from « closed » and put in « open » again.

By definition of algorithms A, the values $f_x(n)$ are calculated using the following formula : $f_x(n) = g_x(n) + h(n)$, where $g_x(n)$ is the *standard term* while h(n) is the *heuristic term*. The *standard function* g is recursively defined hereafter :

Before any extraction (x = 0), only the start node s has appeared. Let $g_0(s)$ be equal to 0.
At the time of the extraction of rank x,
for each son n of node m extracted at rank x
if n has already appeared before extraction x then $g_x(n) = \min(g_{x-1}(n), g_{x-1}(m)+ c(m,n))$
else $g_x(n) = g_{x-1}(m)+ c(m,n)$
for each node n, appeared before extraction x, which is not a son of m, $g_x(n) = g_{x-1}(n)$

Clearly : for any node n, $g_x(n)$ measures the length of a path[1] from s to n in the state graph and $g_x(n)$ decreases (not strictly) when x increases ; thus, when n is fixed, we may consider $g_x(n)$ as a decreasing overestimate of the minimal path length, if any, from s to n (this minimum is denoted g*(n) below). In Fig. 1, $g_{memo}(n)$ keeps track of the minimum of values $g_x(n)$ calculated until now for node n ; likewise, $f_{memo}(n)$ memorizes the minimum of values $f_x(n)$ calculated until now for n ; father(n) records the father node whose expansion led to fix the later value $g_{memo}(n)$. So, any node in « open » is the end of a path from s, whose length is $g_{memo}(n)$ and that only includes expanded nodes ; in addition, any path from s to n whose length is smaller has to include a node not yet expanded.

Heuristic term h(n) only depends on node n at which *heuristic function* h is applied and not of the extraction rank. For this reason, we say that the heuristic function h is *static* while we say that the standard function g (and the evaluation function f) is *dynamic*. Moreover, Nilsson's A algorithm (and most of its successors) supposes that the heuristic h is positive. The measure (according to the length function ; here : $\mathcal{L}_{add}$) of a minimal path, if any, from n to the set T of goal nodes, is denoted h*(n) ; so, h*(s) is the length of a minimal path, if any, from the start node to the set of goals. Commonly, h(n) is considered as an estimate of h*(n).

2.2 Algorithms A*

For Nilsson [21] and other authors[2], algorithms A are designated by A* when the state graph $\mathcal{G}$ and the heuristic function h satisfy the relation : for any node n of $\mathcal{G}$, $h(n) \leq h^*(n)$. We say that h is a *lower-bounding* function[3].

2.3 Admissibility for A*

An algorithm of Heuristic Search in state graphs is called *complete* if and only if it is guaranteed that it terminates finding a path from the start node (s) to the set of goal nodes when such a path exists. An algorithm is called *admissible* if and only if it is complete and the length of the found path is h*(s). The admissibility of A* is proved in [15, 20, 21, 23] supposing that, at once : 1) $\mathcal{G}$ is son-finite and it contains at least one goal, 2) $\exists\, \delta > 0, \forall$ (m,n) arc of $\mathcal{G}$, $c(m,n) \geq \delta$ (this property characterizes the so-called *δ-graphs*), 3) $h \geq 0$. Under the same conditions, algorithms A are complete.

2.4 Admissibility or Sub-Admissibility for Other Algorithms

Later on we shall refer to various successors of Nilsson's A and A*, but without recalling their definitions : Pohl's HPA [25], Pohl's extended A [26], Harris' A [14], Martelli's B [18], Pearl-Kim's A^*_ε [24], Ghallab-Allard A_ε [11, 12], Bagchi-Mahanti's C [1, 2], Pearl's BF* [23, 3, 4], Mero's B′ [19], Korf's IDA* [16], Mahanti-Ray's D [17], Dechter-Pearl's A** [3, 4].

---

[1] For Nilsson, the length of a path $\mathcal{C}$ is the sum of its arc costs. We denote it : $\mathcal{L}_{add}(\mathcal{C})$.
[2] In [20], notation A is not yet used and notation A* does not depend on the fact that « h » is *lower-bounding* or not.
[3] Several authors say that h is *admissible* rather than lower-bounding.

An algorithm is *sub-admissible* if and only if it terminates finding a path from s to T whose length is *near to* h*(s). According to the relations that concretize the « *near to* » notion, the *sub-admissibility* has different meanings. Fig. 2 reports previous results concerning the admissibility of HPA, B, C, IDA*, A** and the sub-admissibility of Harris'A, Pohl's extended A, A_ε^*, A_ε, B′, BF* and D ; for a while, it is sufficient to consider columns 1 and 9.

2.5 Several Interesting Ways to Relax A* and Other Algorithms

Fig. 2 suggests several ways to relax A* and various other algorithms, *while preserving diverse forms of sub-admissibility*.

Constraints About State Graph and Length Functions

Consider columns 2 to 4. All the mentioned algorithms refer to *δ-graphs* (possibly infinite) or finite graphs ; note : in the first case, all the costs are *strictly* positive and in the both cases they are positive[4]. The length of a solution path is always calculated as the *sum* of the costs of its arcs[5]. We shall relax these constraints.

Constraints About Heuristic Functions and Evaluation Functions

Consider column 5 and 6. Most authors associate admissibility with heuristic functions that are at once positive, static and lower-bounding. Nevertheless, we remark that B′, A** and D algorithms can assure minimal solutions using non-static heuristic functions. We observe a form of sub-admissibility for Harris'A and for A_ε while these algorithms use non-lower-bounding heuristic functions. We observe also a form of sub-admissibility for Pohl's extended A while this algorithm uses a non-static and non-lower-bouding heuristic function. Moreover, except for BF* and A**, the evaluation function f is always calculated as the sum (or a linear combination, for HPA) of two functions : the standard function g used by Nilsson's A and a heuristic function (the h of algorithm A or a variant h′). From these examples, we shall propose a more general point of view ; because, in some circumstances, the arc costs (thus the functions f, g, h) may be non-positive (see [10]) ; because, it may be natural to adjust, at least for some nodes n, the quantity h(n) in proportion as the algorithm is running ; because it may be easier to know (or more attractive to use) a non-lower-bounding heuristic function ; because it may be pertinent to examine the interest of other forms of f than a linear combination of g and h or h′.

Constraints About Extraction Modes

Consider column 7. Most algorithms execute *best-first* extractions : as A* does, they systematically extract one of the nodes of open that presently minimize the evaluation function. The A_ε^* and A_ε relax this mechanism. We shall offer more possibilities ; so, the choice of the nodes to expand may be controlled according to a secondary criterion, as previously suggested by [24, 12].

Constraints About Updating Types

Consider column 8. An important (and perhaps undervalued) difference between algorithms lies in the manner to update the values and the pointers associated to each node. The difference appears if we compare the algorithms that use static h with those that use dynamic h : the latter can dissociate the lowerings of g and f. The difference also appears when an algorithm (for instance : BF*) does not exploit the g values : then the pointers are updated if and only if the f values are lowered. We shall develop this topic in Para. 3.1 and 3.3.

---

[4] [10] gives a condition for guarantying admissibility of A* when the state graphs contain non-positive arc costs. We do not know another work dealing with this problem.
[5] Nevertheless the interest of other forms of length is suggested in [27, 23] and illustrated in [30, 6, 13].

1	2	3	4	5	6	7	8	9
Name	Arc costs	Length function	State graphs	Heuristic function	Evaluation function	Extraction mode	Updating type	Length of the discovered path
A*	> 0	$\mathcal{L}$ add	δ-graphs	h : static $0 \leq h \leq h^*$	$g + h$	best-first	1&2&3	$= h^*(s)$
HPA	> 0	$\mathcal{L}$ add	δ-graphs	h : static $0 \leq h \leq h^*$	$(1 - \omega)g + \omega h$ $[0 \leq \omega \leq 1]$	best-first	1&2&3	$= h^*(s)$ [only if $\omega \leq 1/2$]
ext. A of Pohl	≥ 0	$\mathcal{L}$ add	finite graphs	dynamic and positive h′ derived from static h $0 \leq h \leq h^*$	$g_x(n) + h'_x(n) = g_x(n) + h(n) + \lambda(1-\frac{depth_x(n)}{N})h(n)$ $(0 \leq \lambda)$	best-first	3	$\leq (1 + \lambda)h^*(s)$
A of Harris	> 0	$\mathcal{L}$ add	δ-graphs	h : static $0 \leq h$ $h \leq h^* + H$ $(H \geq 0)$	$g + h$	best-first	1&2&3	$\leq h^*(s) + H$
B	> 0	$\mathcal{L}$ add	δ-graphs	h : static $0 \leq h \leq h^*$	$g + h$	best-first	1&2&3	$= h^*(s)$
A^*_ε	> 0	$\mathcal{L}$ add	δ-graphs	h : static $0 \leq h \leq h^*$	$g + h$	n_x if $f_x(n_x) \leq (1+\varepsilon)f_x^*$ $(\varepsilon \geq 0)$	1&2&3	$\leq (1 + \varepsilon)h^*(s)$
A_ε	> 0	$\mathcal{L}$ add	without infinite path whose length is finite	h : static $0 \leq h$ $h \leq (1+\alpha)h^*$ $(\alpha \geq 0)$	$g + h$	n_x if $f_x(n_x) \leq (1+\varepsilon)f_x^*$ $(\varepsilon \geq 0)$	1&2&3	$\leq (1+\varepsilon)(1+\alpha)h^*(s)$
C	> 0	$\mathcal{L}$ add	δ-graphs	h : static $0 \leq h \leq h^*$	$g + h$	best-first	1&2&3	$= h^*(s)$
B′	> 0	$\mathcal{L}$ add	δ-graphs	dynamic and positive h′ derived from static h $0 \leq h \leq h^*$	$g + h'$	best-first	3	$= h^*(s)$
BF*	> 0	$\mathcal{L}$ add but ideas for extension	δ-graphs	dynamic and positive h′ derived from static h $0 \leq h \leq h^*$	$f_x(n) = \Psi(\mathcal{L}(P_{x,n}))$ (Ψ strictly increasing)	best-first	1	$\leq \Psi^{-1}(M)$ (M = min for all paths $\mathcal{P}$ from s to T of $\max_{p \in \mathcal{P}} f(\mathcal{P}_p)$)
A**	> 0	$\mathcal{L}$ add	δ-graphs	dynamic and positive h′ derived from static h $0 \leq h \leq h^*$	$f_x(n) = \max_{p \in P_{x,n}}$ of $\{g_x(p)+h(p)\}$	best-first	3	$= h^*(s)$
IDA*	≥ 0	$\mathcal{L}$ add	finite graphs	h : static $0 \leq h \leq h^*$	$g + h$	best-first	1&2&3	$= h^*(s)$
D	> 0	$\mathcal{L}$ add	δ-graphs	dynamic and positive h′ derived from static and positive h	$g + h'$	best-first	3	= minimal length of paths $\mathcal{P}$ from s to T minimizing $\max_{p \in \mathcal{P}} \{\mathcal{L}\,add(\mathcal{P}_p)+h(p)\}$ [Note: if $h \leq h^*$, this quantity is $h^*(s)$]

Fig. 2. Admissibility/sub-admissibility of diverse Heuristic Search algorithms. $depth_x(n)$: minimal number of arcs from s to n, known at rank x ; N : an upperbound for all x and n. $P_{x,n}$: pointer path of node n at rank x (see Para. 3.8). Updating types : see Para. 3.1 and 3.3. Extraction modes : see Para. 3.4. At rank x, n_x is the extracted node and f_x^* is the min of $f_x(n)$ for all n in open. $\mathcal{P}_p$: the part of $\mathcal{P}$ from s to p. Actually, some results reported in last column were originally established with more restrictive hypothesis not explicited here (it is the case for Harris'A, BF*, A**).

3. Formalizations and Generalizations

Hereafter we present an unifiying formalization. This formalization leads to general theorems, presented in Section 4, which allow to rediscover completely and to extend the table of Fig. 2.

3.1 Algorithms ϱ of Type 1

We call *basic algorithm ϱ for Heuristically-Ordered Search* the code presented in Fig. 3. Later, we shall make the algorithm more precise (to remove the non determinism of the procedures called in lines 4, 8, 9, 10) and we shall consider the application circumstances (what evaluation functions ? what kind of path length ? what state graphs ?). The reference to the evaluation function f is only done in the assignment instructions of the lines 2, 8 and 13 in order to fire 1) an evaluation *now* according to n and according to the execution of the algorithm up to *now*, then 2) an assignment for the variables « $f_{memo}(s)$ », « evolution » and « $f_{memo}(n)$ ».

When a node n appears for the first time (line 13), the evaluation of n is assigned to $f_{memo}(n)$; so for s, in line 2. When n appears again, value $f_{memo}(n)$ may be modified by the update procedure (lines 9 and 10) not yet specified.

According to the manner to make the line 4 more precise, we may define different *extraction modes* (see Para. 3.4). The lines 7 to 15 realize the expansion of the extracted node. Each extracted node that it is not a goal (test in line 5) is immediately *expanded* : *all* its sons are considered (the *partial* expansions are not allowed in this approach). The lines 7 to 15 correspond to the first appearance or eventually a new appearance of the sons of node m lately extracted of open when executing the lines 4 and 6.

```
1   procedure basic-algorithm-ϱ
2       open ← {s} ; closed ← {} ; f_memo(s) ← f_x(s)
3       until open = {} repeat
4       |   m ← extract-node-of-open-according-to-values-f_memo-for-the-nodes-in-open
5       |   if m ∈ T then edit-reverse-path ; stop  endif
6       |   open ← open – {m} ; closed ← closed + {m}
7       |   for each n son of m repeat
8       |   |   evolution ← according-to-characteristics-of-n-including-f_x(n)
9       |   |   if evolution and (n ∈ open) then update endif
10      |   |   if evolution and (n ∈ closed) then  update
11      |   |                                       open ← open + {n}
12      |   |                                       closed ← closed – {n} endif
13      |   |   if n ∉ closed and n ∉ open then  father(n) ← m ; f_memo(n) ← f_x(n)
14      |   |                                    open ← open + {n} endif
15      |   endrepeat
16      endrepeat
17  procedure edit-reverse-path
18      until m = s repeat write m ; m ← father(m) endrepeat ; write "s"
```

Fig. 3. Basic algorithm ϱ

The procedures called in lines 4, 8, 9 and 10 are not yet precisely determined.

The execution of the line 8 assigns « true » or « false » to « evolution » according to whether it is decided or not that value $f_{memo}(n)$, or pointer father(n), must evolve ; this decision may take into account the value of the function f at n *now* , i.e. : $f_x(n)$; but we may also exploit other characteristics of the current node. According to the manner for governing the evolution of $f_{memo}(n)$ and father(n) in lines 8 to 10, we may define different *types* for algorithms ϱ .

Basic algorithm ϱ includes partially determined updating orders, (lines 9 and 10 in Fig. 3). We say that algorithm ϱ is *of type 1* when the updating orders comply with the following rules : at the time of the expansion of any node m and for any son n of m, 1) $f_{memo}(n)$ receives $f_x(n)$ except if the $f_{memo}(n)$ value was already smaller and 2) father(n) receives m except if $f_{memo}(n)$ is not lowered by the step 1.

Remarks about the obtained generalization : Clearly, algorithms A and A* are particular cases of algorithms ϱ of type 1. It may be easily verified that HPA's

Pohl, Harris'A, Martelli's B, Pearl-Kim's A^*_ε, Ghallab-Allard's A_ε, Bagchi-Mahanti's C, Pearl's BF* and Korf's IDA* are also of type 1.

Algorithms ϱ and algorithms ϱ of type 1 do not refer to any cost of arc or path : they may be applied to graphs whose arcs are not valued. When the arcs of the state graph are valued we may be interested in the length of the paths from s to T which are possibly discovered. Nevertheless, if we do not know anything about the relations between the evaluation values of the nodes in open and the length of the paths from s to these nodes, it will be *a priori* difficult to compare the length of the found path with respect to the length of a minimal path. At the contrary, we shall further propose some formulas of sub-admissibility for some algorithms ϱ of type 1 because these algorithms exploit particular evaluation functions. We shall also present some ϱ, called of type 2 or 3, whose codes explicitly refer to costs of arcs and paths. Before, we propose a broadened definition for the *length* of paths.

3.2 Generalization of the Notion of Path Length

Commonly the length of a path is calculated as the sum of the costs of its arcs (see column 3 of Fig. 2). However, Pearl [23] looks at other path lengths, especially the maximal cost of the arcs defining the path. For instance, Yager [30] deals with a family of problems where the path length is calculated as the minimum of the arc costs ; the work described by Dubois, Lang and Prade [6] refers to this family ; Gonella [13] aggregates the arc costs by multiplication. Hereafter we propose a general definition for the lengths of paths. This definition includes the ordinary length $\mathcal{L}_{add}$ and many other particular lengths such as those considered by Pearl, Yager and Gonella ; we shall see in Section 4 that this definition is compatible with interesting properties related to the admissibility or sub-admissibility of Heuristic Search algorithms.

In essence, the operation + and the set $\mathbb{R} =]-\infty,+\infty[$, involved in the definition of the classical length $\mathcal{L}_{add}$, can be respectively replaced by any two place operation Θ and any subset $\mathbb{V}$ of $\mathbb{R}$, provided that : $\mathbb{V}$ is closed under Θ and Θ is associative, increasing in the wide sense (in right and left places) and admits an identity element in $\mathbb{V}$. Now, here is a formal presentation. Let $\mathbb{V}$ be a subset of $\mathbb{R}$ which forms a monoid[6] $(\mathbb{V},\Theta)$ with a two place operation Θ increasing in the wide sense (in right and left places). $\mathbb{C}_\mathbb{V}$ denotes the set of all the sequences of arcs that are valued in $\mathbb{V}$, completed by the empty sequence. c denotes a function which associates to each arc u in $\mathbb{C}_\mathbb{V}$ a value c(u) in $\mathbb{V}$. $\mathbb{C}_\mathbb{V}$ forms a monoid with the operation of concatenation of finite arc sequences. We call *length associated to the monoid* $(\mathbb{V},\Theta)$ *and to the function c*, the function $\mathcal{L}$ whose domain is $\mathbb{C}_\mathbb{V}$, whose range is $\mathbb{V}$ and that respects the following rules : 1) $\forall$ u arc of $\mathbb{C}_\mathbb{V}$, $\mathcal{L}(u) = c(u)$, 2) $\mathcal{L}$ (empty sequence of arcs) = e_n (identity element of the monoid), 3) $\forall\ \mathbb{S}',\mathbb{S}'' \in \mathbb{C}_\mathbb{V}$, $\mathcal{L}(\mathbb{S}'\circ\mathbb{S}'') = \Theta(\mathcal{L}(\mathbb{S}'),\mathcal{L}(\mathbb{S}''))$.

Remarks about the obtained generalization : Taking $\mathbb{V} = \mathbb{R}$ or $\mathbb{R}^+$ or $\mathbb{Q}$ (rational numbers) or $\mathbb{Q}^+$ or $\mathbb{Z}$ (relative numbers), or $\mathbb{N}$ (integer numbers), and $\Theta = +$, we recognize the current forms of the length $\mathcal{L}_{add}$. Other lengths may be made with $\Theta = \times$ and $\mathbb{V} = [0,+\infty[$ or $]0,+\infty[$ or $[1,\infty[$ (also with the restrictions to $\mathbb{Q}$ or $\mathbb{N}$) or $[0,1]$ or $]0,1]$. For instance too, if x and y denote two arc values and if a is a positive number, we can use Θ such as : min(x,y), max(x,y), $x + y - a{\cdot}x{\cdot}y$, $\sqrt{x^2+y^2}$, min (a, x + y), max (0, x + y − a), $x.y/\max(x,y,a)$.

Now, we can define a *generalized standard term* g_x : if $(\mathbb{V},\Theta)$ is the monoid associated to the current length $\mathcal{L}$ then we substitute operation Θ to operation + in the previous definition of g_x (see Para 2.1). It may be verified yet, that for any node n : 1) $g_x(n)$ measures the length $\mathcal{L}(\mathbb{C})$ of a path $\mathbb{C}$ from s to n existing in the state graph and 2) $g_x(n)$ decreases in the wide sense when x increases.

---

[6] $\mathbb{V}$ is closed under Θ, Θ is associative and admits an identity element (e_n) in $\mathbb{V}$.

The minimal path length from s to n (if it exists) will be also denoted $g^*(n)$; the minimal path length from n to T (if it exists) will be also denoted $h^*(n)$.

3.3 Other Types of Updating — Algorithms ϱ of Type 2 or 3

Below we distinguish two types of updating mechanisms that depend on the generalized standard term g. Other updating types could be conceived. We specially consider the types 1, 2 and 3 for the following reasons : 1) all the Heuristic Search algorithms we know use at least one of these updating types[7], 2) for the three types we have obtained results [6, 8] concerning the completeness property, 3) for the type 3 we can present a result[8] concerning the admissibility / sub-admissibility, 4) for the three types we can present a result[9] concerning the length of the possible discovered paths

We shall speak of algorithms *ϱ of type 2* if and only if : at the time of the expansion of any node m and for any son n of m : 1) $f_{memo}(n)$ receives $f_x(n)$ except if the $f_{memo}(n)$ value was already smaller and 2) father(n) receives m except if the value of $g_{memo}(n)$ is not lowered when m is expanded. We shall speak of algorithms *ϱ of type 3* if and only if : at the time of the expansion of any node m and for any son n of m : 1) $f_{memo}(n)$ receives $f_x(n)$ except if the value of $g_{memo}(n)$ is not lowered when m is expanded 2) father(n) receives m except if the value of $g_{memo}(n)$ is not lowered when m is expanded.

Remarks about the obtained generalization : Column 8 of Fig. 2 indicates the updating type of the particular algorithms to which we refer from the beginning. Nilsson's A and A* algorithms are simultaneously of types 1, 2 and 3, because for any node n, the staticity of the heuristic term (function h) leads $f_{memo}(n)$ and $g_{memo}(n)$ to be lowered at the same time. For the same reason, it may be easily verified that Pohl's HPA, Harris'A, Martelli's B, Pearl-Kim's A^*_ε, Ghallab-Allard's A_ε, Bagchi-Mahanti's C and Korf's IDA* are simultaneously of types 1, 2 and 3. Pearl's BF* is a particular algorithm ϱ of type 1. Pohl's extended A, Mero's B′, Mahanti-Ray's D and Dechter-Pearl's A** are particular algorithms ϱ of type 3. We have not met, in the literature of the field, any particular algorithm that is purely of type 2. However, when the updating type is 2, the interesting property of « homogeneity » (defined just below) is satisfied. So, one may suppose that particular algorithms of type 2 will be proposed sooner or later.

Property of « Homogeneity »

For any algorithm ϱ, for any rank of extraction x and for any appeared node n, we call *pointer path of node n at rank x* the path from s to n determined by reversing the sequence n, father(n), father(father(n)),…, s. We denote it $C_{x,n}$. For any updating type, we have : $\mathcal{L}(C_{x,n}) \geq g_x(n)$. For algorithms ϱ of type 2 or 3, it may be easily verified that : $\mathcal{L}(C_{x,n}) = g_x(n)$. We shall say that the *property of homogeneity* is satisfied when for any x rank of extraction, for any n node already appeared *and situated on a path from s to T*, $\mathcal{L}(C_{x,n}) = g_x(n)$. Clearly : algorithms ϱ of type 2 or 3 satisfy the property of homogeneity. *A priori* that is not the case for ϱ's of type 1 ; nevertheless, it may be verified that the specific constraints considered by Pearl imply that its algorithm BF* satisfy the property of homogeneity. This property will be exploited for proving a theorem (Para. 4.5) which provides a general relation concerning the lengths of the solution paths.

3.4 Extraction Modes

Line 4 in Fig. 1 (Nilsson's A) orders a *best-first extraction* that is described in lines 21-22. Many other Heuristic Search algorithms use the same extraction mode.

---

[7] More details are proposed below.
[8] A theorem about the existence and the length of a solution path is given in Para. 4.4.
[9] A theorem concerning the length of a possible solution path is given and proved in Para. 4.5.

The best-first extraction has an intuitive motivation when the algorithms use the evaluations given by g + h in order to guide the search. Indeed, for any node n, $g_x(n)$ may be interpreted as an estimate of g*(n) and h(n) may be interpreted as an estimate of h*(n) ; so $g_x(n) + h(n)$ may be interpreted as an estimate of g*(n) + h*(n), that is to say : the length of a shortest path from s to T passing by n ; finally, the minimal value of $g_x(n) + h(n)$, for any n in « open », may be interpreted as an estimate of the length of a shortest path from s to T ; which leads to respect the « best-first constraint » : extract a node ñ of « open » such as : $f_{memo}(\tilde{n}) = \min_{q \in open} f_{memo}(q)$. However, according to the application context, it may seem not pertinent to choose ñ only by virtue of this formula ; Ghallab-Allard [11, 12] and Pearl-Kim [24, 23] have proposed to relax the « best-first » constraint : given a positive number ε, ñ may be extracted seeing that $f_{memo}(\tilde{n}) \le (1 + \varepsilon) \cdot \min_{q \in open} f_{memo}(q)$; this extraction mode allows to use a secondary criterion, depending on the application, in order to complete the choice of ñ. Hereafter we propose a generalization of the previous extraction modes. We shall see that this generalization is also compatible with interesting properties concerning the lengths ot the solution paths.

Let a function $f \mid x \in D \subset \mathbb{R} \to f(x) \in \mathbb{R}$. We shall say that *f is overpassed on D* when : $\forall x \in D, \exists y \in D, f(x) \le y$. Consider a particular application of an algorithm ϱ. $\mathbb{W}$ denotes the range of the evaluation function f. Suppose that there exists a function $\mathcal{E}$ widely increasing and overpassed on $\mathbb{W}$, such as during this application, for any rank of extraction :

$\mathcal{E}(\min_{q \in open} f_{memo}(q)) \ge \min_{q \in open} f_{memo}(q)$. Suppose that the algorithm systematically extracts a node ñ of « open » such as : $f_{memo}(\tilde{n}) \le \mathcal{E}(f_{memo}(n^*))$.

We shall say that this algorithm uses *$\mathcal{E}$-extraction*

Remarks about the obtained generalization : The best-first extraction is a particular case of $\mathcal{E}$-extraction : take $\mathcal{E}$ equal to the identity function $\mathcal{I}$. Many known algorithms (see column 7 of Fig. 2) use the best-first extraction too : HPA, A of Harris, extended A of Pohl, BF* (note : in « BF* », « BF » means Best-First), B′, D and A**. Algorithms A^*_ε and A_ε use another simple form of $\mathcal{E}$-extraction : take $\mathcal{E} = (1 + \varepsilon) \cdot \mathcal{I}$. It may be verified that algorithms B, C, IDA* use other particular (and less evident) forms of $\mathcal{E}$-extraction.

3.5 Algorithms $\tilde{A}$

In order to define this sub-family of algorithms ϱ we suppose that a length $\mathcal{L}$ is available. Thus there exists a monoid $(\mathbb{V}, \Theta)$ such that Θ is the operation used by $\mathcal{L}$ and any arc of the considered state graphs is valued by a number taken in $\mathbb{V}$.

An algorithm $\tilde{A}$ is an algorithm ϱ whose evaluation function satisfies the following constraint : for any rank of extraction x, for any son n of the node that is extracted at rank x, $f_x(n) = \Theta(g_x(n), h_x(n))$, where $g_x(n)$ is the standard term defined in Para 2.1, while $h_x(n)$ is a value of $\mathbb{V}$ (called *heuristic term*).

For any x and any n, $h_x(n)$ may always be *interpreted* (well knowing or not, according to the context), as *an estimate* of the length of a minimal path, if any, from n to T.

Note : if the monoid $(\mathbb{V}, \Theta)$ is a group, any ϱ may be seen as an algorithm $\tilde{A}$: it is sufficient to take $h_x(n) = \Theta(g'_x(n), f_x(n))$ where $g'_x(n)$ is the inverse of $g_x(n)$ in the monoid $(\mathbb{V}, \Theta)$; if $(\mathbb{V}, \Theta)$ is not a group we may define some ϱ's that are not algorithms $\tilde{A}$: then $\tilde{A}$'s form a *proper* subset of the set made by the ϱ's.

Remarks about the obtained generalization : Nilsson's A is a particular algorithm $\tilde{A}$ that uses a static heuristic term. It may be easily verified that the following algorithms are particular $\tilde{A}$ too : extended A of Pohl, B, C, A^*_ε, A_ε, B′, D. At first sight, the evaluation functions of algorithms BF* and A** have not the form required for algorithms $\tilde{A}$; but BF* and A** are defined on the monoid $(\mathbb{R}^+, +)$ and use evaluation functions f such as : $f \ge g$; thus, the evaluation function f may be formulated as a sum g + h (with $h \geq 0$) and so we can conclude that BF* and A** are particular algorithm Ã. Algorithm HPA behaves as an Ã, because its evaluation function is *proportional* to the evaluation function of an Ã. It may be verified that algorithm IDA* behaves as an Ã *during each iteration*.

We may distinguish algorithms Ã whose updating type is 1, 2 or 3. We call $\tilde{A}_{\mathscr{E}}$ (possibly of type 1, 2 or 3) the Ã's that work with $\mathscr{E}$-extraction.

3.6 More Possibilities for Functions h and f

We propose to relax the constraints commonly required for the evaluation function f and for the heuristic function h. In this less constrained context we propose general results concerning the admissibility or sub-admissibility ; so, it will be possible to use a larger variety of functions f or h and thus the field of applications will be potentially extended. Still we denote by $\mathbb{V}$ the domain of the arc values (a length $\mathscr{L}$ is defined on $\mathbb{V}$, the range of h is included in $\mathbb{V}$), by $\mathbb{W}$ the range of f and by $\mathcal{G}$ the state graph dealt with. Given a Heuristic Search algorithm $\mathscr{A}$ and a state graph $\mathcal{G}$, we denote by $S(\mathscr{A},\mathcal{G})$ the set of the nodes n such that : 1) h*(n) is defined and 2) n is evaluated during the application of $\mathscr{A}$ to $\mathcal{G}$.

Covered h or f

We say that the heuristic function h of an algorithm Ã is *covered by F when the algorithm is applied to* $\mathcal{G}$, if and only if : there exists a function F whose domain is $S(\tilde{A},\mathcal{G})$,whose range is $\mathbb{V}$ and for any node n of $S(\tilde{A},\mathcal{G})$, if n is evaluated at rank x then $h_x(n) \leq F(n)$. We say that the evaluation function f of an algorithm ϱ is *covered by F when the algorithm is applied to* $\mathcal{G}$, if and only if : there exists a function F whose domain is $S(\varrho,\mathcal{G})$,whose range is $\mathbb{V}$ and for any node n of $S(\varrho,\mathcal{G})$, if n is evaluated at rank x then $f_x(n) \leq F(n)$.

Remarks about the obtained generalization : The preceding constraint widely extends the basic constraint respected by Nilsson's A* and many successors : $h(n) \leq h^*(n)$ (in this particular case we said that the *static* heuristic term was « lower-bounding » or « admissible », see Para. 2.2). We can immediately propose several general examples of covered heuristic functions : 1) the h's that are upper-bounded (i.e., $\exists\, M \in \mathbb{V}$, for any node n evaluated at any rank x : $h_x(n) \leq M$), 2) the h's that are *semi-static* (i.e., for any evaluated node n, $h_x(n)$ is allowed to take only a finite number of distinct values), 3) the h's that are simply *static* (i.e., for any evaluated node n, for any i and j ranks for which n is evaluated : $h_i(n) = h_j(n)$), 4) or still the h's that satisfy relations such as : for any node n evaluated at any rank x, $h_x(n) \leq (1 + \alpha)\cdot h^*(n) + H$, where the constants α and H are positive or zero (but note : h is not necessarily static). Now, algorithms A, HPA, B, C, A^*_ε, A_ε and IDA* use static (thus covered) heuristic functions (see column 5 in Fig. 2) ; moreover, A_ε satisfy a relation of the form $h \leq (1 + \alpha)\cdot h^*$, while the particular algorithm A due to Harris satisfy a relation of the form $h \leq H$. It may be verified that the extended A of Pohl uses a semi static (thus covered) h and that the algorithms B′ and D use other particular forms of covered h's. It may be verified that A** use a covered f.

Finitely-decreasing h or f

Given a node n, we denote $[h_x(n)]_x$ the sequence of values taken by the heuristic function h for the successive ranks of extraction x for which n is evaluated. We say that the heuristic function h is *finitely-decreasing when an algorithm Ã is applied to* $\mathcal{G}$, if and only if : for any evaluated node n of $\mathcal{G}$, there does not exist any infinite sub-sequence of $[h_x(n)]_x$ that is strictly decreasing. Likewise, we say that the evaluation function f is *finitely-decreasing when an algorithm* ϱ *is applied to* $\mathcal{G}$, if and only if : for this application, for any evaluated node n of $\mathcal{G}$, there does not exist any infinite sub-sequence of $[f_x(n)]_x$ that is strictly decreasing

Remarks about the obtained generalization : The *finite-decrease* of h or f is a rather soft constraint. Indeed, it seems often natural that for any node n the estimates $h_x(n)$ cannot decrease indefinitely. Obviously, all the semi-static functions and therefore all the static functions are finitely decreasing. It may be easily verified that A, HPA, extended A of Pohl, B, A^*_ε, A_ε, C, B′, IDA* and D use finitely decreasing h (most of them use a static or semi static h ; see Column 5 in Fig. 2).

Quasi-coincident h

We say that heuristic function h is *quasi-coincident when an algorithm $\tilde{A}$ is applied to $\mathcal{G}$*, if and only if : $\exists\, m_T \in \mathbb{V}$, $\exists\, m'_T \in \mathbb{V}$, $\Theta(m_T, m'_T) \geq e_n$ (e_n : identity element), for any evaluated goal node t, $h_x(t) \geq m_T$.

Remarks about the obtained generalization : It may be easily verified that A, HPA, extended A of Pohl, B, A^*_ε, A_ε, C, B′, IDA*, D, SDW are quasi-coincident ; indeed, all these algorithms use $\mathbb{V} = \mathbb{R}^+$, $\Theta = +$ ($e_n = 0$) and a heuristic function h that is positive or zero.

3.7 A Larger Family of State Graphs

We identify now a wide family of state graphs : the $\mathcal{L}$ *-standard* state graphs. It includes, at least, all the state graphs considered in the literature related to A, HPA, extended A of Pohl, B, A^*_ε, A_ε, C, BF*, B′, IDA*, D and A**.

$\mathcal{L}$ -uncompressibility

Let $\mathcal{L}$ be the used length[10]. We say that a state graph $\mathcal{G}$ is $\mathcal{L}$ *-uncompressible* if and only if : $\forall\, M \in \mathbb{V}$, $\exists\, k \in \mathbb{N}$, $\forall\, \mathcal{C}$ elementary[11] path of $\mathcal{G}$ from s, $\mathcal{N}(\mathcal{C}) > k \Rightarrow \mathcal{L}(\mathcal{C}) > M$, where $\mathcal{N}(\mathcal{C})$ denotes the number of arcs of $\mathcal{C}$.

Remarks about the obtained generalization : Clearly : any δ-graph (see Para 2.3) is $\mathcal{L}_{add}$-uncompressible and any finite graph is $\mathcal{L}$ -uncompressible (whatever is $\mathcal{L}$). All the algorithms above-mentioned, except A_ε, have been analyzed by their authors in case of application to δ-graphs (see Column 4 in Fig. 2) ; Ghallab and Allard have defined and studied A_ε when it is applied to graphs less constrained than δ-graphs, but these graphs are still particular $\mathcal{L}_{add}$-uncompressible graphs.

$\mathcal{L}$ -standard State Graphs

We say that a state graph is $\mathcal{L}$ *-standard* if and only if it satisfies the four following conditions : it is son-finite, it contains at least a goal, it is $\mathcal{L}$ -uncompressible, it does not contain any absorbant circuit[12]. It may be easily verified that any $\mathcal{L}$ -standard graph owns at least one minimal path (in the sense of $\mathcal{L}$) from s to T.

Remarks about the obtained generalization : All the above-mentioned algorithms have been analyzed by their authors in case of application to particular $\mathcal{L}$ -standard graphs.

4. General Results to Approach the Sub-admissibility

Within the preceding formal framework, we can prove various general results. So, the five original properties presented below form a set of tools to establish formulas of sub-admissibility. The arguments of the proofs are given in [8]. For general results about the *termination finding a path to a goal*, see [9]. Let $\mathcal{G}$ be a goal-accessible state graph at which an algorithm $\tilde{A}$ is applied. It is easily verified that : at the time of any extraction i, on any path of $\mathcal{G}$ from s to T, there exists a node belonging to « open » whose predecessors along the path belong to « closed »; this node is called : *input of the path for the i^{th} extraction*.

4.1 Lemma

Let $\mathcal{G}$ be a state graph at which is applied an algorithm $\tilde{A}$ of type 1, 2 or 3 whose heuristic term h is covered by F . Let $\mathcal{C}$ be a path of $\mathcal{G}$ from s to T such as, $\forall$ i rank of extraction, $g_{i-1}(e_i) = \mathcal{L}(\mathcal{C}_{e_i})$ and $h^*(e_i)$ is defined, where e_i is the input of

---

10 Length relative to some monoid ($\mathbb{V}$,Θ) ; the arc values are taken in $\mathbb{V}$ (see Para. 3.1).
11 i.e. : without repeating any state.
12 Absorbant circuit : whose length < e_n (identity element of the monoid of arc values).

$\mathcal{C}$ at the time to decide the i^{th} extraction and $\mathcal{C}_{e_i}$ is the part of $\mathcal{C}$ from s to e_i. Then : $f_i(e_i) \leq \Theta(\mathcal{L}(\mathcal{C}_{e_i}),F(e_i))$, where $f_i(e_i)$ is the evaluation value fastened to e_i at the i^{th} extraction. This result generalizes « lemma 3.1 » proposed[13] in [20] (also : « result 2 » in [21] and « lemma 1 » in [23]).

4.2 Theorem

Let $\mathcal{G}$ be a state graph without absorbant circuit. Suppose that Θ is strictly increasing in the left place[14]. Let an algorithm ϱ of type 3 applied to $\mathcal{G}$. Let n be any node of $\mathcal{G}$. Let $\mathcal{C}$ be any minimal path from s to n. Denote $\mathcal{C} = n_0, \ldots, n_t$ where $n_0 = s$ and $n_t = n$. $\forall$ i rank of extraction, if $n \notin$ closed, let n_k ($k \in \mathbb{N}$, $0 \leq k \leq t$) be the input of $\mathcal{C}$ at the time to decide the i^{th} extraction. Then : $\forall j \in \mathbb{N}$, $0 \leq j \leq k \Rightarrow g_{i-1}(n_j) = g^*(n_j)$. This result generalizes the « lemma 2 » proposed[15] in [23].

4.3 Theorem

Let $\mathcal{G}$ be a state graph without absorbant circuit and that contains at least one goal. Let an algorithm $A_{\mathcal{E}}$ of type 3 applied to $\mathcal{G}$, which terminates. Suppose that, when the algorithm is applied, heuristic function h is covered by F and quasi-coincident (thus : $\exists m_T \in \mathbb{V}$, $\exists m'_T \in \mathbb{V}$, $\Theta(m_T,m'_T) \geq e_n$). Suppose that Θ is strictly increasing in the left place. Then, at the time of the termination, the algorithm has found a path $\mathcal{C}$ from s to T such as : $\mathcal{L}(\mathcal{C}) \leq \Theta(\mathcal{E}(\Theta(g^*(e_i),F(e_i))),m'_T)$, where e_i is the input, when the i^{th} and last extraction is decided, of any minimal path from s to T in $\mathcal{G}$. In the following theorem, the termination of the algorithm is not an hypothesis but a part of the conclusion.

4.4 Theorem of Sub-Admissibility

Let $\mathcal{G}$ be a $\mathcal{L}$-standard state graph at which is applied an algorithm $\tilde{A}_{\mathcal{E}}$ of type 3, whose heuristic function h is covered by F during the application, quasi-coincident (thus : $\exists m_T \in \mathbb{V}$, $\exists m'_T \in \mathbb{V}$, $\Theta(m_T,m'_T) \geq e_n$) and lower-bounded by an element m of $\mathbb{V}$ such as $\exists m' \in \mathbb{V}$, $\Theta(m',m) \geq e_n$. Suppose that Θ is strictly increasing in the left place. Then, the algorithm terminates finding a path $\mathcal{C}$ of $\mathcal{G}$ from s to T such as : $\mathcal{L}(\mathcal{C}) \leq \Theta(\mathcal{E}(\Theta(g^*(e_i),F(e_i))),m'_T)$, where e_i is the input, at the time to decide the i^{th} and last extraction, of any minimal path from s to T in $\mathcal{G}$.

This result generalizes « theorem 3.1 » proposed[16] in [20] (also : « result 4 » in [21] and « theorem 2 » in [23]).

4.5 Theorem of the Found Path

Let $\mathcal{G}$ be a state graph that contains at least one goal. Let an algorithm ϱ of type 2 or 3, that is applied to $\mathcal{G}$ and terminates at the time of the i^{th} extraction ; let $extract_i$ be the i^{th} extracted node. Suppose that there exists a function $\boldsymbol{\Omega}$, whose domain is the range T_f of $f_x(n)$ when n runs on T such as : for any evaluated goal node t, $g_x(t) \leq \boldsymbol{\Omega}(f_x(t))$. Then, at the time of the termination the algorithm has found a path $\mathcal{C}$ from s to T such as : $\mathcal{L}(\mathcal{C}) \leq \boldsymbol{\Omega}(f_i(extract_i))$. This theorem holds for the algorithms ϱ of type 1 when the property of homogeneity is satisfied.

This result generalizes the « theorem 2* » proposed by Pearl[17] in [23].

---

13 Statement of « lemma 3.1 » in [20] : « If $\hat{h}(n) \leq h(n)$ for all n, then at any time before A* terminates and for any optimal path P from node s to a goal, there exists an open node n′ on P with $\hat{f}(n') \leq f(s)$ ».

14 For instance, this property is true if $(\mathbb{V},\Theta)$ is a group rather than a simple monoid.

15 Statement of « lemma 2 » in [23] : « Let n′ be the shallowest OPEN node on an optimal path $P^*_{s-n''}$ to any arbitrary node n″, not necessarily in Γ. Then : $g(n') = g^*(n')$, stating that A* has already found the optimal pointer-path to n′ (i.e., n′ is along $P^*_{s-n''}$) and that path will remain unaltered throughout the search ».

16 Statement of « theorem 3.1 » in [20] : « If $h(n) \leq h(n)$ for all nodes n, and if all arc costs are greater than some small positive number δ, then algorithm A* is admissible ».

17 Statement of « theorem 2* » in [23]: « BF* is $\Psi^{-1}(M)$-admissible, that is, the cost of the solution path found by BF* is at most $\Psi^{-1}(M)$ ».

Remarks about some rediscoveries and generalizations : Nilsson's A* algorithm is also a particular case of $\tilde{A}_{\mathcal{E}}$ of type 3 ; its heuristic function h is covered by F = h*. The state graphs considered by Nilsson contain at least one goal and do not admit absorbant circuits. The heuristic function is quasi-coincident (take $m_T = m'_T = 0$). The operation Θ (that is to say +), is strictly increasing in the left place. Thus we may apply theorem 4.4 : $\mathcal{L}_{add}(\mathcal{C}) \leq g^*(e_i) + h^*(e_i)$, where $\mathcal{L}_{add}(\mathcal{C})$ is the length of the found path $\mathcal{C}$ from s to T while e_i is the final input of any minimal path $\mathcal{C}_0$ from s to T[18]. Because e_i belongs to $\mathcal{C}_0$ (minimal) : $g^*(e_i) + h^*(e_i) = \mathcal{L}_{add}(\mathcal{C}_0)$. Thus $\mathcal{L}_{add}(\mathcal{C}) = h^*(s)$: the admissibility of A*'s is thus rediscovered. The admissibility may be still proved when some constraints applied to Nilsson's A* are relaxed ; indeed, providing that the updates remain of type 3, theorem 4.4 is yet applicable to monoids $(\mathbb{V}, \Theta)$ other than group $(\mathbb{R}^+, +)$, to state graphs that are not necessarily δ-graphs, to heuristic functions h that are not necessarily static or positive. Moreover, theorem 4.4 gives some formulas of sub-admissibility for extraction modes which may be not best-first and for h which may be not lower-bounding. If we relax Nilsson's A* algorithms towards $\tilde{A}$'s of type 1 or 2, we can establish some formulas of sub-admissibility by combining the theorem 4.5 with the general results about completeness given in [9]. Likewise, we have applied[19] theorems 4.1 to 4.5 for extending the known results about the admissibility or the sub-admissibility concerning HPA, extended A of Pohl, B, A^*_ε, A_ε, C, BF*, B′, IDA*, D and A**.

5. Concluding Remarks and Perspectives

We have proposed a formalization concerning the Heuristically-Ordered Search in state graphs. We have considered 5 dimensions : 1) the notion of length to measure the paths between nodes, 2) the characteristics of the state graphs dealt with, 3) the choices of the nodes to expand, 4) the kinds of updating to realize, 5) the properties of the evaluation functions that guide the search. We have employed this formalization to present several general theorems about the admissibility or sub-admissibility. These theorems allow to extend the availability of Heuristic Search algorithms (yet existing or not) ; thus, the field for applications is potentially enlarged. Moreover, the proposed analysis allows an unifying and comparative presentation (just suggested here by lack of space) of various classical algorithms ; this analysis facilitates a better understanding of the key points, the limitations and the non exploited potentialities.

We have supposed that the cost of a move from any node to another one is a real number ; it may be interesting to extend our formalization for dealing with other kinds of values, namely fuzzy numbers or fuzzy intervals (see chapter 5 of [7] and [5] for a basic study of Heuristic Search in an imprecise environment). Our work may be also developed in order to tackle the problems of completeness, admissibility or sub-admissibility for other variants of Heuristic Search algorithms (such as real-time algorithms, restricted-memory algorithms or algorithms for dynamic environnements) or to extend the results relative to the bidirectional algorithms (see [8], chapter 8, for preliminary work).

References

1. A. Bagchi and A. Mahanti, Search algorithms under different kinds of heuristics — A comparative study, *J. ACM* 30 (1) (1983) 1-27.
2. A. Bagchi and A. Mahanti, Three approaches to heuristic search in networks, *J. ACM* 32 (1) (1985) 1-27.
3. R. Dechter and J. Pearl, Generalized best-first search strategies and the optimality of A*, *J. ACM* 32 (3) (1985) 505-536.

---

[18] For any $\mathcal{L}$-standard graph, there exists a minimal path from s to T.

[19] We don't give the details by lack of place.

4. R. Dechter and J. Pearl, The optimality of A*, in: L. Kanal and D. Kumar, eds, *Search in Artificial Intelligence* (Springer-Verlag, 1988) 166-199.
5. D. Dubois, H. Farreny and Henri Prade, Combinatorial search with fuzzy estimates, in : J. Kacprzyk and S. A. Orlowski, eds, *Optimization models using fuzzy sets and possibility theory* (D. Reidel publishing company, 1987) 171-185.
6. D. Dubois, J. Lang and H. Prade, Theorem proving under uncertainty. A possibility theory-based approach, in : *Proceedings 10th IJCAI*, Milan, Italy (1987) 984-986.
7. D. Dubois and H. Prade, *Possibility theory - An Approach to Computerized Processing of Uncertainty*, (Plenum, 1988).
8. H. Farreny, *Recherche Heuristiquement Ordonnée — Algorithmes et propriétés*, (Masson, Paris, 1995).
9. H. Farreny, Une généralisation pour la Recherche Heuristiquement Ordonnée : les algorithmes ρ et la propriété d'arrêt avec découverte de solution, in : *Proceedings of RFIA 96*, Rennes, France (1996) 225-234.
10. D. Gelperin, On the optimality of A*, *Artificial Intelligence* 8 (1) (1977) 69-76.
11. M. Ghallab, Optimisation de processus décisionnels pour la robotique, Thèse de Doctorat d'Etat, Université Paul Sabatier, Toulouse, France (1982).
12. M. Ghallab and D. G. Allard, A_ε : An efficient near admissible heuristic search algorithm, in : *Proceedings 8th IJCAI*, Karlsruhe, Germany (1983) 789-791.
13. R. Gonella, Diagnostic de pannes sur avions : mise en oeuvre d'un raisonnement révisable, Thèse de l'École Nationale Supérieure de l'Aéronautique et de l'Espace, Toulouse, France (1989).
14. L. R. Harris, The heuristic search under conditions of error, *Artificial Intelligence* 5 (3) (1974) 217-234.
15. P. E. Hart, N. J. Nilsson and B. Raphael — A formal basis for the heuristic determination of minimal cost paths, *IEEE Trans. SSC* 4 (1968) 100-107.
16. R. E. Korf, Depth-first iterative-deepening : an optimal admissible tree search, *Artificial Intelligence* 27 (1985) 97-109.
17. A. Mahanti and K. Ray, Network search algorithms with modifiable heuristics, in : L. Kanal and D. Kumar, eds, *Search in Artificial Intelligence*, (Springer-Verlag, 1988) 200-222.
18. A. Martelli, On the complexity of admissible search algorithms, *Artificial Intelligence* 8 (1) (1977) 1-13.
19. L. Mero, A heuristic search algorithm with modifiable estimate, *Artificial Intelligence* 23 (1) (1984) 13-27.
20. N. J. Nilsson, *Problem-solving methods in artificial intelligence*, (Mc Graw-Hill, 1971).
21. N. J. Nilsson, *Principles of artificial intelligence*, (Tioga, 1980).
22. J. Pearl, Some recent results in heuristic search theory, *IEEE Trans. on PAMI* 6 (1) (1984) 1-12.
23. J. Pearl, *Heuristics : intelligent search strategies for computer problem solving* (Addison-Wesley, 1984).
24. J. Pearl and J. H. Kim, Studies in semi-admissible heuristics, *IEEE Trans. on PAMI* 4 (4) (1982) 392-400.
25. I. Pohl, First results on the effect of error in heuristic search, in : B. Meltzer and D. Michie, eds, *Machine Intelligence* 5 (Edinburgh University Press, 1969) 219-236.
26. I. Pohl, The avoidance of (relative) catastrophe, heuristic competence, genuine dynamic weighting and computational issues in heuristic problem solving, in : *Proceedings 3d IJCAI*, Stanford, USA (1973) 12-17.
27. M. J. Schoppers, On A* as a special case of ordered search, *Proc. 8th IJCAI*, Karlsruhe, Germany (1983) 783-785.
28. H. A. Simon, Search and Reasoning in Problem Solving, *Artificial Intelligence* 21 (1-2) (1983) 7-29.
29 B. Steward, C. F. Liaw and C. C. White III, A bibliography of Heuristic Search Research Through 1992, *IEEE transactions on Systems, Man, and Cybernetics* 24 (2) (1994) 268-293.
30. R. R. Yager, Paths of least resistance in possibilistic production systems, *Fuzzy sets and systems* 19 (1986) 121-132.