Des heuristiques plus performantes si avec la fonction on garde la raison

Henri Farreny
ENSEEIHT-INP de Toulouse / Institut de Recherches en Informatique de Toulouse (IRIT)
118 Route de Narbonne — 31062 Toulouse — France — Email : Henri.Farreny@irit.fr

Résumé

On présente une vue synthétique et critique du rôle des heuristiques dans la résolution des problèmes de taquins n x n. On constate que les heuristiques sont couramment exploitées comme des fonctions numériques seulement. On montre qu'il est possible de prendre en considération d'autres connaissances significatives, ordinairement négligées. On propose une méthode qui équivaut à engendrer des heuristiques présentant a priori une forme de supériorité par rapport à celles disponibles auparavant. On rapporte diverses expérimentations, sur les taquins, qui confirment l'intérêt pratique et théorique de cette approche.

Mots-clefs

Heuristiques - Recherche Ordonnée - IDA* - A* - Taquins

Abstract

We present a synthetical and critical view concerning the part played by heuristics in n x n puzzles. We observe that heuristics are commonly used only as numerical functions. We show that it is possible to take into consideration other significant knowledge, commonly neglected. The method that we propose is equivalent to generate some heuristics that satisfy a form of superiority with respect to those previously available. We report various experimentations, with eight-puzzle and fifteen-puzzle, that confirm the practical and theoretical interest of this approach.

Keywords

Heuristics - Ordered Search - IDA* - A* - Eight-puzzle - Fifteen-puzzle.

1 Vue d'ensemble

Notre motivation est d'améliorer les performances dans la résolution des problèmes de taquins et plus largement de tirer un meilleur parti des heuristiques disponibles. L'article comporte un volet synthèse et un volet innovation. Le domaine d'inspiration et d'application est d'abord celui des taquins classiques, mais l'idée principale nous semble pouvoir servir pour d'autres applications.

Dans le premier volet, on introduit la problématique des taquins. On montre que l'étude des taquins se nourrit de progrès conceptuels, méthodologiques et technologiques, en informatique au sens large, et en stimule de nouveaux. On met l'accent sur le rôle des heuristiques. On caractérise brièvement mais précisément les algorithmes de base afin de faciliter la compréhension de la suite.

Dans le deuxième volet on fait valoir que les heuristiques sont couramment exploitées comme des fonctions numériques exclusivement. On montre qu'il est possible — et plausiblement souhaitable — de prendre en considération d'autres connaissances significatives, ordinairement négligées. On propose à cet effet une méthode qui revient à engendrer des heuristiques présentant a priori une forme de supériorité par rapport à celles disponibles auparavant. On rapporte diverses expérimentations comparatives sur les taquins 3 x 3 et 4 x 4, avec A* et IDA* , qui confirment nettement l'intérêt pratique de l'approche : moins d'espace et moins de temps consommés. L'idée principale peut être appliquée à d'autres heuristiques que celles que nous avons étudiées pour les problèmes de taquins et à d'autres problèmes que les taquins.

2 Problématique des taquins : bilan

Trouver des solutions minimales pour des problèmes quelconques de taquins classiques ("n x n puzzle") reste un problème difficile. Les - modestes - progrès accomplis depuis 40 ans tiennent au développement des modèles et algorithmes de la Recherche Heuristiquement Ordonnée, à l'accroissement de la puissance de calcul et de mémorisation des ordinateurs, et à l'exploitation de nouvelles heuristiques. Notre thèse ici est qu'on peut mieux tirer parti des heuristiques qu'en les réduisant à des fonctions numériques. Pour pouvoir l'exposer de manière tangible nous avons pris le parti de faire un tour d'horizon synthétique : nous pourrons ainsi mieux mesurer les difficultés, apprécier les évolutions et situer notre contribution.

2.1 Un vieux problème qui résiste

Les taquins sont des jeux devenus populaires à la fin du siècle dernier et encore très familiers aujourd'hui. Dans la forme matérielle la plus répandue (cf fig. 1), 15 jetons carrés, de même taille, portant chacun une marque exclusive (ici les nombres de 1 à 15) sont juxtaposés à l'intérieur d'un cadre carré qui pourrait accueillir 16 jetons de ce type, rangés sur 4 lignes et 4 colonnes. L'absence d'un 16e jeton ménage une "case vide" vers laquelle peut glisser n'importe quel jeton voisin (2, 3 ou 4 voisins). Le jeu consiste[1] à déplacer les jetons un par un pour transformer un "état initial" quelconque en un "état-but" fixé, en un nombre minimal de coups. Par exemple, il suffit de 9 coups pour transformer l'état initial s_1 de la fig. 1 en l'état-but t (et on ne peut pas faire moins). Par contre il faut (et il suffit de) 80 coups pour transformer s_2 en t. Ce genre de taquins est dit "taquin 4 x 4" ("Fifteen-puzzle" ou "4 x 4 puzzle").

1	5	2	3
4	6		7
8	14	10	11
12	9	13	15

s_1

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

t (but)

15	11	8	12
14	10	9	13
2	6	1	4
3	7	5	

s_2

Figure 1. Problème facile de taquin 4 x 4 : passer de s_1 à t.
Problème très difficile : passer minimalement de s_2 à t.

La simplicité des composants et des règles du jeu est particulièrement attrayante. Mais on constate vite que bien jouer n'est pas chose simple ! Une première difficulté, qui fut à l'origine de la "taquin mania" des années 1890-1900, vient du fait que de nombreux problèmes de taquin 4 x 4 ne sont pas résolvables. Deuxième difficulté : dès qu'on enchaîne quelques dizaines de coups, à partir de n'importe quel état initial, on a du mal à gérer la multitude d'états différents qui se présentent ; or l'expérience montre que les solutions, quand on

[1] Dans cette étude on ne s'intéresse qu'à la production de solutions *minimales*. Produire une simple solution est beaucoup moins épineux (pour un ordinateur ou un humain). Produire une "bonne" solution est un autre problème encore.

parvient à en trouver, dépassent souvent la cinquantaine de coups. Troisième difficulté : lorsqu'on découvre une solution il est difficile d'assurer qu'elle est minimale en nombre de coups[1], ou même "pas trop longue". Ces difficultés, immédiatement sensibles au grand public, qu'en sait le milieu scientifique ?

On pourrait se borner à déclarer tout de go que Ratner et Warmuth ont prouvé en 1986 que trouver une solution minimale pour un problème quelconque de taquin n x n est NP-dur [26]. Heureusement, en pratique, on n'affronte pas le taquin n x n dans toute sa splendeur (n quelconque) : on peut rencontrer de nombreux cas intéressants avec n fixé (voire plafonné). Et quelle que soit la complexité qui pèse a priori, on peut avoir besoin de résoudre tel cas précis ou telle famille restreinte de cas. Le résultat de Ratner et Warmuth n'est donc pas une "fin de non-étudier" pour les taquins n x n : il laisse la porte ouverte pour analyser (formellement et / ou expérimentalement) des problématiques moins ambitieuses que celle affrontée par eux.

2.2 Taquins 4 x 4 : un monde mal connu

On peut associer à chaque état d'un taquin 4 x 4 un sommet de graphe et à chaque coup possible une arête entre 2 sommets. Le graphe ainsi défini (dit "graphe d'états") compte 16! sommets (20 922 789 888 000). Jouer équivaut à chercher dans le graphe d'états une chaîne entre les sommets représentant l'état initial et l'état-but.

Dès 1879, Johnson et Storey [13] ont établi un "caractère de résolvabilité" facile à mettre en oeuvre[2], grâce auquel il apparaît que le graphe du taquin 4 x 4 comporte 2 composantes connexes comptant chacune $16! / 2 \approx 10^{13}$sommets. En conséquence, si on tire au sort l'état initial on a exactement 1 chance sur 2 de tomber sur un problème non résolvable et on peut tester immédiatement si tel est le cas.

Mais, lorsqu'on sait qu'un problème est résolvable, on ne connaît pas de formule pour engendrer une solution ou seulement fournir la longueur minimale d'une solution.

Jusqu'à récemment on ignorait la valeur du maximum des longueurs des solutions minimales de tous les problèmes résolvables ; ce nombre, qui mesure le maximum des longueurs (comptées en nombres d'arcs) des chaînes minimales entre tout couple de sommets appartenant à une même composante connexe est le "diamètre" du graphe d'états. En 1997, Brungger [1][2] a établi que le diamètre du graphe du taquin 4 x 4 est soit 80 soit 81.

2.3 Taquin 3 x 3 : une terre cultivée

L'humble taquin 3 x 3 ("eight-puzzle") a été le plus complètement exploré. En 1967, Schofield [29] a publié une étude exhaustive d'une version simplifiée : les cases vides doivent être au centre (2e ligne, 2e colonne) aussi bien dans l'état initial et dans l'état-but . En 1993, Reinefeld [27] a traité tous les problèmes de taquin 3 x 3 "à but standard" : la case vide est en haut à gauche, tandis que les jetons 1 à 8 sont disposés dans cet ordre, de gauche à droite et de haut en bas. Pour chacun des 16! / 2 problèmes résolvables il produit les solutions de longueur minimale (non seulement 1 mais toutes). Il établit que le maximum des solutions minimales vaut 31 (et que 2 problèmes seulement exigent autant de coups) ; en montrant que les solutions des problèmes à but non standard ne peuvent être plus longues, on conclut que le diamètre du graphe du taquin 3 x 3 est 31.

Cependant, même pour le taquin 3 x 3 il se peut qu'il reste des propriétés à découvrir quant aux rapports *a priori* entre problèmes, longueurs minimales et nombres de solutions.

2.4 Autres taquins : une forêt vierge

Pour les taquins n x n (n x n - 1 jetons du même type juxtaposés sur n lignes et n colonnes ("n x n puzzle"), on peut aussi montrer qu'il existe 2 composantes connexes comptant chacune (n x n) ! / 2 sommets. Mais on ignore le diamètre des graphes des taquins n x n dès que n > 4.

Et il existe bien d'autres sortes de taquins : taquins m x n, taquins avec plus d'une case vide, taquins dont le cadre de jeu n'est plus rectangulaire ni même convexe, taquins enveloppant un cylindre ou une sphère, taquins à jetons de formes inégales (on connaît souvent "L'âne rouge")... Les taquins font partie d'une vaste famille de solitaires : "Sliding-Tile Puzzles", qui inclut le cube de Rubik. La plupart de ces jeux sont vierges d'analyse mathématique ou informatique (résolvabilité, diamètre, méthodes...).

2.5 Un défi pour l'informatique au sens large

La maîtrise des taquins pourrait être directement utile pour traiter des problèmes pratiques de parcage, magasinage, circulation (automobiles, wagons, marchandises, matériaux, informations...).

Mais c'est d'abord sur les plans conceptuel et méthodologique que l'intérêt d'étudier les taquins est le plus patent. Les taquins intéressent du point de vue de l'intelligence artificielle certes mais aussi de l'algorithmique générale [3], de la théorie des graphes [20], de la combinatoire [32], de l'algorithmique parallèle [2].

Plus particulièrement, depuis une quarantaine d'années, les taquins (les plus "simples") accompagnent le développement de la Recherche Heuristiquement Ordonnée (RHO), une des branches maîtresses de l'Intelligence Artificielle. Nous présentons maintenant, en substance (principes, propriétés), deux outils de base de la RHO : les algorithmes A* et IDA*, en vue de faciliter la présentation des extensions que nous proposons et expérimentons ensuite.

3 Taquins et RHO : apports mutuels

L'étude de Schofield citée plus haut [29] a été réalisée au milieu des années 60. En conjuguant des considérations algébriques, des astuces de représentation et une procédure de type "en largeur d'abord", l'auteur parvient à résoudre les 8! / 2 problèmes résolvables du taquin restreint considéré mais ne traite pas le taquin 3 x 3 proprement dit, faute de ressources (mémoire, temps, bon algorithme). Pour aller plus loin il fallait de nouvelles ressources.

3.1 Au commencement était A*...

En 1968, Hart, Nilsson et Raphael [12] proposent l'algorithme A* (voir aussi [21][22][23][24]). A* est un algorithme de la famille RHO : partant d'un état initial (noté s dans la suite) il cherche à découvrir un état-but (t dans la suite) en guidant l'explicitation progressive du graphe d'états au moyen d'une "fonction d'évaluation" f.

[2] Chaque état étant représenté par la suite des numéros des jetons (par ex. : de haut en bas et de gauche à droite, case vide numérotée 0), un problème peut-être résolu si et seulement si l'état-but est une permutation paire de l'état initial.

A* "développe" d'abord s, c'est-à-dire engendre la totalité de ses fils (4, 3 ou 2 selon la position de la case vide) ; puis il développe celui des fils p dont l'évaluation f(p) est la plus petite. Les états développés constituent la "réserve". Les états engendrés nouveaux alimentent le "front". A* suit le principe dit du "meilleur d'abord" (best-first) : systématiquement il développe l'un des états du front qui présente la plus petite évaluation ; il s'arrête dès que l'état ainsi sélecté se révèle être l'état-but t ; si en développant un état on constate qu'un de ses fils p est déjà dans le front ou la réserve on compare la valeur de f(p) *maintenant* à la valeur f(p) qui était *connue jusqu'ici* et on garde la plus petite ; si f(p) est abaissée pour un p qui est dans la réserve on le transfère dans le front.

La spécificité de A* au sein de la famille RHO alors naissante réside principalement dans les caractéristiques exigées pour la fonction d'évaluation f. Elle doit être de la forme f = g + h ; pour chaque état p, g(p) est la longueur du chemin le plus court couramment connu joignant l'état initial s à p (pour les fils de s, g vaut tout de suite 1; pour un descendant lointain de s on peut disposer successivement de plusieurs valeurs décroissantes) ; h(p) est une valeur qui ne dépend que de p (on dit que h est "statique" ; on peut interpréter h(p) comme une estimation de la longueur minimale de chemin depuis p jusqu'à t ; on réserve souvent à h l'appellation de (composante) "heuristique" de f). Hart, Nilsson et Raphael prouvent[3] que si h est "minorante" (c'est-à-dire : si, $\forall$ p $h(p) \leq h^*(p)$, où $h^*(p)$ est la longueur du chemin le plus court — absolu, éventuel, généralement inconnu — joignant p à t) alors A* est "admissible", c'est-à-dire : s'il existe un chemin de s à t, A* en trouve un minimal.

3.2 Premier cadre formel pour heuristiques

Ils définissent aussi une relation d'ordre partiel entre heuristiques : si h_1 et h_2 sont 2 heuristiques minorantes relatives au même graphe d'états, h_2 est "mieux informée" que h_1 si et seulement si $\forall$ état p distinct de t on a $h_1(p) < h_2(p)$. Et ils prouvent que : tous les états développés par A* muni de h_2 sont développés par A* muni de h_1 ("théorème d'optimalité" ou de comparaison, voir [21][22][24][23][5]).

En outre, si une heuristique est "monotone" (c'est-à-dire : si, $\forall$ p, $\forall$ p' fils de p, $h(p) \leq c(p,p') + h(p')$, où c(p,p') est le coût de l'arc de p vers p' ; pour les taquins classiques tous les coûts d'arcs valent 1), alors tout état ne peut être développé qu'une fois ; ce résultat permet une comparaison stricte entre développements accomplis par A* muni de 2 heuristiques dont l'une est mieux informée que l'autre et de surcroît monotone.

3.3 P : heuristique populaire

A l'époque où est conçu A* on connaît 2 heuristiques minorantes pour taquins : W et P (en sus de la fonction nulle et des dérivés simples de W et P tels que leurs combinaisons convexes). Pour un état p, W(p) est le nombre de jetons qui ne sont pas à leur place ; par exemple, pour la fig. 1, $W(s_1) = 6$, $W(s_2) = 15$. P(p) dénote la "distance de Manhattan"[4] de p ; par exemple, pour la fig. 1, $P(s_1) = 7$, $P(s_2) = 58$. W et P sont toutes deux mieux informées que la fonction nulle. Toutes deux sont monotones aussi. Bien que P ne soit pas formellement mieux informée que W ($\exists$ état p distinct de t tel que W(p) = P(p)), c'est évidemment elle qui est préférée[5].

Pendant la quinzaine d'années qui suivent son invention, l'algorithme A* inspire de nombreuses variantes et / ou alternatives exploitant des heuristiques minorantes (notamment : algorithmes B de Martelli, C de Bagchi et Mahanti, BF* de Pearl, B' de Mero, D de Mahanti et Ray) ou "quasi" minorantes (notamment : algorithmes HPA de Pohl, A de Harris, $A\varepsilon$ de Ghallab, $A\varepsilon^*$ de Pearl et Kim, SDW de Köll et Kaindl). Divers algorithmes 'bidirectionnels" dotés d'heuristiques minorantes ou quasi minorantes sont conçus.

Pour des compléments et des références complètes on peut voir : [24][5][14][30][6][7][23].

Les qualités formelles de la plupart des algorithmes qui ont succédé à A* sont généralement mises à l'épreuve par expérimentation sur le taquin 3 x 3 (rarement plus) avec P pour heuristique.

Néanmoins il a fallu l'avènement de l'algorithme IDA*, pour que, encore avec P comme auxiliaire, quelqu'un (Reinefeld [27]) entreprenne de résoudre tout taquin 3 x 3 (sur une Sun Sparc Station).

3.4 Enfin IDA* vint...

A* et nombre de ses succédanés présentent le défaut d'être gourmands en mémoire : ils maintiennent en mémoire tous les états créés. Par exemple, lorsque A*, muni de P, résout le problème de la fig. 1 (solution minimale produite : 9 coups), il exécute 18 développements et garde, au fur et à mesure, les 45 états créés[6] (à la fin, |front| = 27 et |réserve| = 18). En 1985 Korf rapporte qu'ayant tiré au sort 100 problèmes[7], de taquin 4 x 4 et tentant de leur appliquer A*, muni de P, sur un DEC 2060, il ne parvient à résoudre aucun d'entre eux : pour les 100 cas, malgré une programmation soignée, il constate un dépassement de capacité-mémoire après création d'environ 30 000 états. C'est ce qui l'amène à concevoir l'algorithme IDA* (Iterative-Deepening A*) [15][23].

IDA* utilise une fonction d'évaluation f qui a les caractéristiques déjà exigées par A* ; par contre, IDA* ne suit pas le principe "développer un état de *plus petite évaluation d'abord*" mais : "développer *partiellement* l'état le *plus profond d'abord* pourvu que son évaluation ne dépasse pas le seuil alloué courant S". Initialement S vaut h(s). IDA* produit un premier fils p de s, puis un premier fils de p et ainsi de suite tant que l'évaluation de l'état le plus profond est inférieure ou égale à S ; si cette évaluation dépasse S, IDA* remonte au carrefour de choix immédiatement précédent — soit q l'état correspondant — et engendre un nouveau fils de

---

[3] Avec quelques restrictions, satisfaites par les taquins.

[4] $\forall$ état p et $\forall$ jeton j, col(j) et lig(j) sont respectivement les nombres de colonnes et lignes qui séparent j dans p de j dans t. P(p) est la somme des col(j) et lig(j) étendue à tous les j.

[5] L'expérience confirme que P est beaucoup plus efficace que W.

[6] Nombres liés à l'hypothèse suivante : pour développer tout état on considère d'abord le fils obtenu en déplaçant la case vide vers le haut puis la gauche, puis la droite, puis le bas.

[7] Nous baptisons $K_{4,1}$ à $K_{4,100}$ les 100 exemples de Korf en commençant par celui (n° 79 dans [15]) pour lequel Korf a visité le moins d'états (540 860) jusqu'à celui (n° 88) pour lequel il en visite le plus (6 009 130 748). L'état-but est standard : case vide en haut à gauche, puis jetons 1 à 15 de gauche à droite et de haut en bas.

q. S'il ne reste plus de choix, IDA* lance une nouvelle passe d'exploration en profondeur, depuis s, mais en assignant maintenant à S le minimum des valeurs d'évaluation ayant dépassé la valeur en vigueur jusqu'ici. IDA* s'arrête lorqu'il advient que le sommet à développer est l'état-but .

Avec les mêmes hypothèses pour f et h que celles retenues pour A*, Korf montre[8] que IDA* est admissible (et même que IDA* ne développe que des états développés aussi par A*). L'énorme avantage de IDA* sur A* c'est qu'à tout instant il ne conserve en mémoire que les états d'un chemin joignant s à l'état le plus profond du moment, chemin de longueur majorée par h*(s) ; si, par hypothèse ordinaire, il existe un minorant d > 0 du coût de tous les arcs (par exemple, pour les taquins n x n : d = 1), le nombre d'états de ce chemin est majoré par h*(s)/d ; IDA* est donc de complexité linéaire en ce qui concerne l'espace requis.

En contre-partie, chaque fois que le seuil est augmenté, IDA* entreprend une nouvelle passe d'exploration en profondeur au cours de laquelle il refait tout ou partie du travail accompli à la passe précédente (tout si la nouvelle passe n'est pas la dernière), ce qui peut prendre beaucoup de temps.

En 1985 donc, grâce à IDA* muni de h = P, Korf détermine une solution minimale pour chacun des problèmes $K_{4,1}$ à $K_{4,100}$; la résolution sur DEC 2060 consomme environ une demi-heure par problème ; la longueur moyenne des solutions minimales est environ 53. Mais quand il s'agit d'attaquer des problèmes quelconques de taquin 5 x 5, IDA* muni de h = P est beaucoup trop lent[9].

17	1	20	9	16
2	22	19	14	5
15	21		3	24
23	18	13	12	7
10	8	6	4	11

s

	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

t

Figure 2. Problème $K_{5,1}$: passer de s à t minimalement. Solution minimale : 100 coups.

En 1996 Korf et Taylor expérimentent avec IDA* des heuristiques *supérieures* à P (nous préciserons au § 3.5) ; ils réussissent à trouver des solutions minimales pour 9 problèmes de taquin 5 x 5 parmi une série de 10 tirés au hasard (notés ci-après $K_{5,1}$ à $K_{5,10}$; l'état-but est standard). Ils travaillent sur une station Sun Ultra Sparc (70 fois plus rapide que la DEC 2060). Le problème le plus vite résolu ($K_{5,1}$, cf fig. 2 ; solution minimale : 100 coups) consomme 2 h et quart de temps de calcul ; le plus lentement résolu ($K_{5,9}$; solution minimale : 113 coups) consomme 1 mois ; la résolution de $K_{5,10}$ a été interrompue après 1 mois de calcul. La longueur moyenne des 9 solutions minimales trouvées est environ 101 ; pour $K_{5,10}$ la solution minimale est assurée ≥ 112. A notre connaissance personne n'a fait mieux en ce qui concerne les solutions minimales pour taquins n x n, n ≥ 5.

3.5 Recours à des heuristiques supérieures à P

Pour aborder le taquin 5 x 5, Korf et Taylor ont employé une nouvelle heuristique dont nous reprenons les composantes principales, sous l'appellation P&LC&C, avec la définition :

P&LC&C, = P + LC + C.

P est la distance de Manhattan.

Voici les principes déterminant LC et C.

8	7	6
5	4	3
2	1	

s_1

	1	2
3	4	5
6	7	8

t

3	1	4
	2	5
8	7	6

s_2

Figure 3. Deux problèmes de taquin 3 x 3 : Passer minimalement de s_1 à t et de même de s_2 à t.

LC (Linear Conflict Heuristic) est décrite en détails par Hanson et coll. [11] mais on en trouve déjà mention chez Gardner [9]. Si dans l'état p, 2 jetons sont dans la rangée (ligne ou colonne) qui leur est destinée dans t, mais qu'ils sont en ordre inverse, il faudra nécessairement que l'un des 2 jetons quitte la rangée pour "laisser passer" l'autre puis y revienne. On note que ces deux coups supplémentaires ne sont pas considérés lorsqu'on évalue P (ils ne pèsent rien pour P). Par exemple, en fig. 3, l'état s_1 présente plusieurs conflits en ligne 2 et colonne 2 ; naturellement il faut gérer les rapports entre conflits pour ne pas surévaluer leur coût[10].

C (Corner-Tiles Heuristic) est décrite par Korf [18]. Dans l'état s_2 de la fig. 3, le jeton 4 occupe 1 coin qui ne lui est pas destiné ; mais le jeton 1 ("voisin de coin"), lui, est à sa place ; pour que le jeton 4 puisse quitter le coin et que le jeton 2 puisse y arriver, il faudra que 1 bouge, ce qui coûtera 2 coups au moins (pour aller et venir). Korf compte de même pour 3 coins sur 4 : il ne tient pas compte du coin haut-gauche (pour éviter des tests complexes). Sur un taquin n x n avec n > 3, ce système peut peser jusqu'à 12 coups ; 6 coups seulement sur un taquin 3 x 3 car un voisin d'angle ne doit pas intervenir 2 fois. En outre, si on a déjà pris en compte la contribution de LC, il faut éviter de recompter des coups supplémentaires pour les jetons voisins d'angle déjà impliqués dans des conflits linéaires ; ainsi, pour l'état s_2 de la fig. 3, le jeton 7 est impliqué dans un conflit sur la ligne 3.

Avec les précautions signalées quant aux interférences entre ses composantes, P&LC&C est minorante. En outre : ∀ état p, P(p) ≤ P&LC&C(p). Au sens strict de Nilsson et coll., P&LC&C n'est pas mieux informée que P : on ne peut donc invoquer le théorème de comparaison. Mais l'expérience confirme l'attente intuitive : de manière générale, P&LC&C conduit à moins de développements et, malgré le surcoût d'évaluation, consomme moins de temps ; des indications numériques seront présentées dans les tableaux 5 et 6. Nous dirons simplement que P&LC&C est *supérieure* à P. La suite montrera (§ 5) que nous pourrons donner à cet adjectif une acception qui n'est pas seulement numérique.

Nous emploierons aussi dans la suite l'heuristique minorante notée P&LC, définie par : P&LC = P + LC.

[8] Avec quelques restrictions, satisfaites par les taquins. Dans [6][7][8] il est montré que ces restrictions (et celles usuellement associées à A*) peuvent être largement relaxées : A* et IDA* relèvent d'une famille générale dotée de bonnes propriétés.

[9] Notion relative, bien sûr, selon le support matériel.

[10] _Indiquons qu'ici, il ne faut pas compter LC(s_1) = 12.

3.6 Force-brute et ingéniosité combinées

Avec IDA*, le taquin 3 x 3 a été sensiblement maîtrisé, le taquin 4 x 4 est devenu abordable, le taquin 5 x 5 a été approché (N.B. : toujours du point de vue de la recherche de solutions *minimales*). Pour les taquins 3 x 3 et 4 x 4, l'heuristique P suffisait (quoique LC et C eussent été déjà bienvenues pour le 4 x 4). Ces progrès sont allés de pair avec l'emploi de machines plus puissantes.

L'un des défauts de IDA*, contre-partie de son extrême sobriété en espace, c'est qu'il ne met pas à profit les capacités en mémoire des ordinateurs d'aujourd'hui. IDA* tend à répéter de nombreux calculs.

Diverses améliorations algorithmiques ont été imaginées, notamment : MREC de Sen et Bagchi, DFS* de Rao, Kumar et Korf, BIDA* de Manzini, DBIDA* et BDBIDA* d'Eckerle. Des améliorations de performances parfois très importantes (BIDA* spécialement) ont été réalisées bien que l'heuristique-reine demeure P.

Culberson et Schaeffer [4] ont ouvert une nouvelle voie : celle des "pattern databases" qui table sur la disposition d'une forte puissance de mémorisation pour exploiter des bases préétablies d'informations heuristiques.

Pour le taquin 4 x 4, ils utilisent deux heuristiques minorantes. Voici le principe de la première. Pour atteindre l'état-but t depuis p il faut constituer tôt ou tard la colonne de droite et la ligne du bas ("the fringe") à l'image de t (jetons 3, 7, 11, 15, 12 13 14 pour le but t standard). Imaginons que le but soit relaxé : il s'agit seulement maintenant d'amener au plus vite les jetons 3, 7, 11, 15, 12 13 14 à leur destination dans t, sans se préoccuper des cases où parviennent les autres jetons (qui sont donc traités comme indistincts entre eux). Notons fringe(p) la longueur de la solution minimale ; évidemment fringe est minorante. En focalisant leur attention sur une autre forme ("the corner"), les auteurs définissent une autre heuristique minorante, corner. En prenant max(fringe corner) on obtient l' heuristique minorante qui sera finalement employée. On constitue (une fois pour toutes : but standard) une base de données BDF qui fait correspondre à toute configuration possible des jetons 3, 7, 11, 15, 12 13 14 la valeur de fringe. Et de même BDC pour corner.

La résolution d'un problème quelconque combine IDA*, muni de P (encore !), avec la consultation de BDF et BDC. En appliquant cette technique à $K_{4,1}$ - $K_{4,100}$, le nombre d'états visités est divisé par 346 en moyenne et le temps par 6.

Korf [19] a proposé récemment plusieurs développements de cette technique.

Mais personne n'a mené à bien, pour le taquin 4 x 4, une campagne systématique à la Reinefeld.

Il a fallu attendre 1997 pour qu'il soit établi, en recourant à des calculateurs parallèles de grande puissance, que le maximum des solutions minimales du taquin 4 x 4 (avec but standard) valait 80 et que soient identifiés les 17 problèmes de taquin qui exigent ce nombre de coups pour être résolus.

Brungger et ses collègues [1][2] ont mobilisé pendant plusieurs semaines un Intel Paragon et un NEC Cenju-3 disposant de 64 à 128 processeurs (au total environ 1300 jours d'unité centrale répartis sur ce système).

Ils ont pris appui sur le travail de Gasser [10] qui avait isolé des problèmes pour lesquels 80 coups étaient nécessaires. Ils ont utilisé une heuristique majorante ("iterative fringe heuristic") : ils écartent tous les problèmes pour lesquels cette heuristique donne un majorant de solution minimale en-dessous de 80 coups.

Au prix d'environ 700 jours d'unité centrale ils ont ainsi isolé un ensemble d'environ 37 000 problèmes susceptibles d'exiger plus de 80 coups. Au prix d'environ 600 jours d'unité centrale, ils ont appliqué une variante de IDA*, munie de P seulement, pour chercher une solution minimale à chacun de ces problèmes. Tous les problèmes sauf 17 étaient résolvables en moins de 80 coups. Les 17 autres exigeaient exactement 80 coups.

Corollairement, Brungger a prouvé que le diamètre du graphe du taquin 4 x 4 est soit 80 soit 81.

4 Heuristiques : savoir raison garder

Instinctivement, tout un chacun apprécie le "poids" des heuristiques : entre 2 heuristiques minorantes dont l'une est supérieure à l'autre (au sens indiqué plus haut) on a tendance à préférer celle qui est "plus proche" de h*. Si l'une est mieux informée que l'autre (au sens indiqué plus haut), c'est encore mieux : on bénéficie du théorème de comparaison, éventuellement renforcé si l'heuristique mieux informée est monotone.

Le "poids" d'une heuristique importe bien sûr. Entre deux heuristiques comparables ou non. Par rapport aussi au terme g de la fonction d'évaluation (de nombreux travaux ont porté sur ce sujet). Mais la capacité de discernement d'une heuristique ne se limite pas à la grandeur de ses valeurs.

Certains auteurs ont mis l'accent à juste titre sur "l'origine" des heuristiques : [24][11][18][25]. La réflexion à ce sujet accompagne (mais ne précède pas toujours !) certains progrès méthodologiques.

Dans la suite nous voulons aller plus au fond : mieux capter les connaissances attachées à une heuristique. Au delà de la simple distribution des valeurs (qui a rapport au poids qu'on lui attribue), au-delà de la raison d'une heuristique (qui a rapport à son origine), nous voudrions compléter l'exploitation des valeurs par celle des raisons.

4.1 Au-delà de la simple valeur...

La plupart des nombreuses publications qui rapportent des expériences sur taquins n x n (avec très généralement n =3 ou 4 rappelons-le) utilisent l'heuristique P. Elle est même encore largement utilisée alors que d'autres *supérieures*, telles P&LC, ont été publiées … sans doute parce qu'elle est simple à calculer.

Quoi qu'il en soit, pour un état p, on n'utilise de P(p) que sa *valeur*, cette valeur est considérée intéressante, d'une part parce qu'elle est minorante (i.e. $P(p) \leq h^*(p)\ \forall p$) et d'autre part parce qu'elle est une assez haute (sous-) estimation de h* (ainsi : nettement supérieure en général à W(p)). La minorance de P assure l'admissibilité des algorithmes A* ou IDA* (et dérivés) qui y recourent. Sa proximité à h* contribue à discerner entre les états susceptibles de conduire à une solution minimale et ceux qu'il n'est pas la peine de développer.

Nous proposons d'exploiter non seulement la valeur de P mais aussi le système de contraintes qui est à l'origine de cette valeur.

4.2 P-résolvabilité

Par exemple, en fig. 1 la valeur $P(s_1) = 7$ indique qu'il faut au moins 7 coups pour joindre p à l'état-but ; mais ce nombre ne capture pas toute l'information dont nous disposons : nous savons *en outre* que l'état-but sera effectivement joignable en 7 coups si et seulement si 1 jeton, le 9, effectue 1 pas vers le haut, et si 1 jeton, le 13, effectue 1 pas la gauche, et si 3 jetons, les 1-6-14, effectuent 1 pas vers la droite, et si 2 jetons, les 5-14, effectuent 1 pas vers le bas. Nous pouvons tester cette possibilité en lançant depuis p une exploration en profondeur d'abord, avec profondeur limitée à la valeur $P(s_1) = 7$ et en n'autorisant que les mouvements compatibles avec les contraintes précédentes.

Si, cette exploration, sous ces contraintes, échoue c'est qu'il faut au moins 1 coup qui éloigne au moins 1 jeton de sa destination finale et au moins 1 coup en sens inverse (pour ramener ce jeton).

Donc, si l'exploration *locale* et *restreinte* (*locale* depuis p, *restreinte* à P(p) coups satisfaisant un ensemble précis de contraintes) n'atteint pas p on peut affirmer qu'un chemin depuis p vers l'état-but compte au moins P(p) + 2 coups. Alors nous posons : $P^f(p) = P(p) + 2$ (P^f résume l'idée : "éprouver P *forward*"). Si l'exploration atteint l'état-but, nous posons $P^f(p) = P(p)$; dans ce cas-là nous dirons que p est P-résolvable.

4.3 Heuristique P^f

Par construction, l'heuristique P^f est minorante tout en étant *supérieure à* P (mais pas *mieux informée* au sens de Nilsson). La minorance de P^f assure que A* ou IDA* (éventuellement d'autres algorithmes RHO) guidés par $f = g + P^f$ seront admissibles ; la supériorité de P^f ne garantit pas, a priori, des performances meilleures mais on peut l'escompter raisonnablement et le vérifier expérimentalement.

Pourquoi l'escompter ? Parce que P^f discerne entre les états qui sont plus loin du but que ne l'estime P et ceux qui sont exactement à la distance prévue : P^f leste les premiers de 2 coups supplémentaires.

Pour A*, qui compare les évaluations entre elles, le classement des états candidats au développement est changé (au bénéfice des états pour lesquels P dit vrai).

Pour IDA*, qui compare les évaluations au seuil courant, le développement de certains états est autorisé quand on se fie à P mais interdit quand on se fie à P^f.

Certes l'évaluation de P(p) se trouve alourdie par le supplément d'exploration depuis p ; mais les développements engagés depuis p se font *sans* évaluation et avec un degré de branchement en gros divisé par 2 (2 directions pour les coups au plus au lieu de 4 au plus).

Nous avons testé l'heuristique P^f, d'abord pour résoudre 200 problèmes de taquin 3 x 3 tirés au hasard[11], puis pour résoudre les 10 premiers problèmes[12] de Korf pour 4 x 4 ($K_{4,1}$ - $K_{4,10}$).

	développements	états mémorisés	états créés sans évaluation	temps
P	1 045	1 612	0	1*
P^f	320	508	2 235	0,117
P^f+	310	487	1 611	0,105

Tableau 1. Résultats moyens obtenus en appliquant A* à 200 problèmes 3 x 3 tirés au sort. Le temps d'exécution avec P sert de référence (1* représente environ 12,25 s sur notre Macintosh G4)

Dans la partie supérieure du tableau 1 on compare les performances de A* muni de P à celles de A* muni de P^f.

Pour les 200 problèmes considérés, on constate que le nombre moyen d'états mémorisés (via front et réserve) est divisé par environ 3,3 quand on emploie P^f au lieu de P.

Le temps moyen d'exécution[13] est divisé par près de 10.

	états visités	passes	créés sans évalu.	temps
P	3 454	5,1	0	1*
P^f	980	4,1	7 048	0,84
P^f+	964	4,1	5 672	0,71

Tableau 2. Résultats moyens obtenus en appliquant IDA* aux mêmes 200 problèmes 3 x 3. Le temps d'exécution avec P sert de référence (1* représente environ 2 s sur notre Mac G4)

Dans la partie supérieure du tableau 2 on compare IDA* muni de P à IDA* muni de P^f.

Pour les 200 problèmes considérés, (les mêmes qu'avec A* précédemment), on constate que le nombre moyen d'états visités est divisé par 3,5 quand on emploie P^f au lieu de P ; le temps moyen d'exécution est abaissé de 16 %.

L'amélioration est intéressante mais bien moindre que celle observée pour A*.

	états visités	passes	créés sans évalu.	temps
P	1 117 438	6,8	0	1*
P^f	182 813	5,8	1 830 955	0,56
P^f+	182 785	5,8	1 518 489	0,48

Tableau 3. Résultats moyens obtenus en appliquant IDA* aux problèmes $K_{4,1}$ à $K_{4,10}$. Le temps d'exécution avec P sert de référence (1* représente environ 16 mn 10 s sur notre Mac G4).

Le tableau 3 concerne un échantillon de problèmes de taquin 4 x 4 : les problèmes $K_{4,1}$ à $K_{4,10}$.

Dans la partie supérieure, on compare IDA* muni de P à IDA* muni de P^f. Le nombre moyen d'états visités est divisé par 6,1 ; le temps moyen d'exécution est abaissé de 42 %.

Nous ne rapportons pas d'éxpérience relative à P et P^f avec A* sur les problèmes $K_{4,1}$ à $K_{4,10}$ car nous nous heurtons à des problèmes d'espace et de temps sur notre matériel[14].

4.4 Ajustements pour mieux tirer parti de P^f

[11] La moyenne des solutions minimales est 22,18.

[12] La moyenne des solutions minimales est 45,2.

[13] Toutes les exécutions rapportées ici ont eu lieu sur Macintosh G4 cadencé à 350 MHz (code écrit en Lisp). Les temps sont comparés en *proportion* par rapport à une exécution de référence sur ce même ordinateur (par exemple A* avec P).

[14] Rappelons que c'est parce qu'il ne parvenait pas à résoudre ces problèmes avec A* et P que Korf a été conduit à définir IDA*.

Jusqu'ici nous testions P^f avec les algorithmes A* et IDA* standard. Mais il convient d'adapter légèrement ces algorithmes pour mieux tirer parti de la nouvelle heuristique.

Premièrement, lorqu'un état p apparaît P-résolvable (donc nous posons $P^f(p) = P(p)$) nous le marquons comme tel ; supposons qu'un état q est choisi pour être développé, soit par A* soit par IDA*, alors qu'il possède la marque de P-résolvabilité ; rappelons que A* et IDA* standards maintiennent chacun un chemin C(s,q) joignant l'état-initial s à q, de longueur g(q) ; soit C(q,t) le chemin joignant q à l'état-but t qui est découvert lorsque le test de P-résolvabilité de q réussit ; sa longueur est P(q) ; soit C(s,t) la concaténation de C(q,t) derrière C(s,q) ; la longueur de C(s,t) est g(q) + P(q), c'est-à-dire f(q) ; rappelons (cf [23] par exemple) que les algorithmes A* et IDA* satisfont la propriété commune suivante : quel que soit l'état q qui est choisi pour être développé, $f(q) \leq h^*(s)$; or $h^*(s)$ est la longueur d'un chemin minimal joignant s à t ; donc $f(q) = h^*(s)$ et C(s,t) est minimal de s vers t.

Donc on n'a pas besoin de développer les sommets P-résolvables : quand on en rencontre un (disons q) un chemin minimal est disponible (Cs,t).

Deuxièmement, il n'est pas nécessaire de tester la P-résolvabilité d'un état p' dont le père courant p a été reconnu non P-résolvable, dès lors que P(p') < P(p).

En effet, on peut affirmer que p' est non P-résolvable car : les contraintes qu'il faudrait respecter jusqu'au but pour tester la P-résolvabilité de p' sont un sous-ensemble de celles dont le respect jusqu'au but eut prouvé la P-résolvabilité de p.

Dans les dernières lignes des tableaux 1, 2 et 3, intitulées "P^f +" on illustre l'effet des ajustements de A* et IDA* que nous venons de décrire.

Comme prévu les performances sont meilleures que celles obtenues avec A* standard et IDA* standard.

On constate une réduction des nombres d'états créés non évalués et, en conséquence, une réduction supplémentaire appréciable des temps d'exécution.

Sur l'échantillon de 200 problèmes 3 x 3, l'ajustement de A* (voir tableau 1, ligne P^f +) procure un gain en temps, par rapport à A* standard muni de P^f, de 11 % tandis que l'ajustement de IDA* (voir tableau 2, ligne P^f +) procure un gain en temps de 13,8 % par rapport à IDA* standard muni de P^f. Sur l'échantillon de 10 problèmes 4 x 4, l'ajustement de IDA* (voir tableau 3, ligne P^f +) procure un gain en temps de 14,3 %.

5 Une méthode générale pour mieux exploiter les heuristiques

Nous étendons maintenant à d'autres heuristiques les techniques que nous avons introduites et testées avec P. Pour la présente communication les expériences resteront limitées à des problèmes de taquins n x n (champ déjà important et riche comme soutenu depuis le début). Nous continuons à analyser ces techniques pour exploitation par A* et IDA*, sans préjudice de l'intérêt qui pourrait être porté à d'autres algorithmes RHO.

On a pu observer qu'un certain nombre de fonctions minorantes employées comme heuristiques trouvent leur origine (ou s'interprètent a posteriori) en relâchant des règles du problème originel [24][11][25][18][19]. Mais on peut renverser la façon de voir : revenant au problème originel on peut essayer de satisfaire les contraintes ignorées dans le problème relaxé. Si on n'y parvient pas on dispose d'une information supplémentaire.

5.1 Contraintes intrinsèques à une heuristique

W est parfaite pour le *problème relaxé* où chaque jeton peut être immédiatement posé à la place qui lui est destinée, qu'elle soit occupée ou non.

En revenant au *problème originel*, on observe que pour joindre p à t en exactement W(p) coups il faut et il suffit de respecter les contraintes suivantes : un coup quelconque (légal pour le problème originel) est jouable si et seulement si le jeton bougé atteint illico sa destination.

Parce qu'elles sont intrinsèques à W et paramétrées par p et t nous notons ces contraintes $CI_W(p,t)$.

De même la fonction P est parfaite pour le *problème relaxé* où chaque jeton peut librement se rapprocher de sa destination, à raison d'une case par coup, sans tenir compte des obstacles. En revenant au *problème originel*, on observe que pour joindre p à t en exactement P(p) coups il faut et il suffit de respecter les contraintes suivantes : un coup quelconque (légal pour le problème originel) est jouable si et seulement si il rapproche le jeton joué de sa destination. Nous notons $CI_P(p,t)$ ces contraintes.

On pourra donc conclure pour le problème originel (non relaxé) que : la fonction W sous-estime strictement les chemins qui séparent p de t si et seulement si il est impossible de joindre p à t en W(p) coups respectant $CI_W(p,t)$. De même : la fonction P sous-estime strictement les chemins qui séparent p de t si et seulement si il est impossible de joindre p à t en P(p) coups respectant $CI_P(p,t)$.

5.2 h-résolvabilité pour h quelconque

De façon générale, soit h une fonction minorante telle qu'on puisse affirmer que "la fonction h sous-estime strictement les chemins qui séparent p de t si et seulement si il est impossible de joindre p à t en h(p) coups qui respectent chacun un jeu de contraintes noté $CI_h(p,t)$".

Nous dirons que p est "h-résolvable" si et seulement si h(p) coups, respectant $CI_h(p,t)$, *suffisent* pour joindre p à t (par ailleurs h(p) coups sont *nécessaires* dès lors que h est minorante).

Inversement, p n'est pas h-résolvable si et seulement si il faut au moins 1 coup supplémentaire.

5.3 Heuristiques r-punitives

Notons que dans le cas particulier où h était P, nous avons exploité la propriété : "il faut au moins 1 coup supplémentaire par rapport à ceux exigés par P" implique "il faut au moins 2 coup supplémentaire par rapport à ceux exigés par P" ; car un jeton qui s'éloigne de sa destination devra, pour atteindre t, dépenser tout autant pour annuler l'éloignement.

Nous dirons que h est r-punitive si, quels que soient p et t, ne pas respecter $CI_h(p,t)$ impose au moins r coups supplémentaires. Ainsi, P mais aussi P&LC et P&LC&C sont 2-punitives.

5.4 Heuristique supérieure engendrée depuis h : h^f

Étant donné une heuristique h r-punitive, nous pouvons définir h^f, "heuristique supérieure engendrée depuis h après test en avant", comme suit : $h^f(p) = h(p)$ si p est h-résolvable, sinon $h^f(p) = h(p) + r$.

La fonction h^f est minorante si et seulement si h l'est. Nous pouvons l'employer avec A*, IDA* et d'autres algorithmes RHO, pour estimer, à la place de h, la distance séparant p de t. Pour calculer $h^f(p)$ chaque fois que nécessaire nous greffons sur A* ou IDA* un "module de test de h-résolvabilité" ; il s'agit d'un générateur de coups à qui l'on spécifie qu'il part de p, qu'il ne doit pas dépasser la distance h(p) depuis p et qui est astreint à respecter CIhp,t).

Disposant de h(p) comme profondeur d'exploration et vu l'intérêt de gérer dynamiquement l'espace exploré, nous optons pour un générateur travaillant en profondeur d'abord. Si ce générateur en profondeur atteint t en h(p) coups, la minorance de h implique que h(p) = h*(p) ; alors le générateur a découvert un chemin minimal, soit C(p,t), de p à t.

5.5 Avancement du test d'arrêt

On établit aisément (reprendre la formalisation présentée plus haut concernant les états P-résolvables, en substituant h à P) qu'il n'y a pas lieu de développer les états h-résolvables : si un tel état p est choisi, par A* ou IDA*, pour être développé, alors un chemin minimal de s à t, passant par p, est immédiatement disponible en concaténant le chemin C(s,p) qui supporte g(p) au chemin C(p,t) révélé par le test de h-résolvabilité.

5.6 Heuristiques cohérentes

Nous dirons que h est "cohérente" si pour un fils quelconque p' d'un état quelconque p obtenu en respectant CIhp,t), tout coup depuis p' qui satisfait CIhp',t) satisfait nécessairement CIhp,t). Il est évident par exemple que P est cohérente ; on peut établir aussi que P&LC et P&LC&C, sont cohérentes.

Attention a priori la cohérence n'est pas à confondre avec la monotonie au sens défini plus haut.

5.7 Élimination de tests de h-résolvabilité inutiles

Étant donné une heuristique h cohérente, il n'est pas nécessaire de tester la h-résolvabilité d'un état p' dont le père courant p a été reconnu non h-résolvable, dès lors que h(p') < h(p). En effet, dans ces conditions, on peut affirmer a priori que p' est non h-résolvable.

Sinon, depuis p' on pourrait atteindre t en h(p') coups respectant CIhp',t) ; h(p') < h(p) implique que p' respecte CIhp,t) ; Alors la cohérence de h implique que tout coup respectant CIhp',t) respecte aussi CIhp',t) ; donc depuis p on peut atteindre t en h(p') + 1 coups, c'est-à-dire au plus h(p) coups, respectant h(p) ; donc p' est h-résolvable : contradiction.

5.8 Expériences avec P&LC^f P&LC&C^f

Les heuristiques P&LC, P&LC&C étant minorantes et 2-punitives, nous pouvons définir, respectivement, les "heuristiques supérieures engendrées depuis P&LC et P&LC&C après test en avant", que nous notons : P&LC^f et P&LC&C^f.

Ci-dessous nous présentons des résultats obtenus en appliquant A* ou IDA* munis de P&LC^f ou P&LC&C^f.

Sans entrer dans les détails, attirons l'attention sur quelques problèmes subalternes.

Pour concrétiser le passage d'une heuristique à l'heuristique supérieure h^f engendrée par h, il faut implémenter un module de test de h-résolvabilité et gérer des informations spécifiques pour ce test.

Dans le cas de l'heuristique P, le module de test est relativement simple, car à tout instant au cours du test il suffit de référer au but t pour connaître les mouvements encore autorisés.

Pour LC, la tâche est plus délicate ; voici quelques indications sur la manière dont nous procédons ; supposons qu'un conflit linéaire entre 2 jetons i et j est retenu (donc pèsera 2 coups dans le calcul de LC) ; à cet instant on sait que l'un ou l'autre des 2 jetons sera exceptionnellement autorisé à quitter la rangée de conflit pour laisser passer l'autre ; par exemple ils pourront, l'un *ou* l'autre (ou exclusif) exécuter 1 coup vers le haut *ou* vers le bas (ou exclusif encore) ; en conséquence, nous plaçons le couple (i j) dans une pile LCh où sont tenus à jour les "mouvements spéciaux vers le haut résultant de LC" et de même, dans une pile LCb où sont tenus à jour les "mouvements spéciaux vers le bas résultant de LC" ; lors du test de LC-résolvabilité, ou de P&LC résolvabilité, ou de P&LC&C-résolvabilité, cette pile sera consultée ; si l'un ou l'autre des jetons i ou j utilise 1 coup spécial vers le haut par exemple, le couple (i j) sera retiré de LCh et LCb ; LCh, LCb (et LCd, LCg) sont gérés comme des piles par le module de test, pour se conformer à sa stratégie de génération de coups en profondeur d'abord.

De même, pour implémenter le module de C-résolvabilité, nous gérons une pile qui identifie à chaque instant les jetons qui sont autorisés à des coups spéciaux correspondant aux contraintes intrinséques de l'heuristique C.

La gestion de ces piles, quoique de petite taille, suppose un travail supplémentaire. On peut sans doute améliorer notre implémentation actuelle pour gagner en temps.

A* ou IDA* peuvent être adaptés pour intégrer l'avancement du test d'arrêt (avancement pertinent dès qu'on utilise une heuristique supérieure engendrée depuis une autre).

Parce que P&LC et P&LC&C sont cohérentes, A* ou IDA* peuvent aussi être adaptés pour éliminer des tests inutiles de, respectivement, P&LC-résolvabilité ou P&LC&C-résolvabilité.

Lorsque ces 2 adaptations de A* ou IDA* sont prises en compte, le nom de l'heuristique est suivi du signe +.

Dans le tableau 4 l'effet général de notre technique est très appréciable. Le nombre de développements est divisé par près de 3 quand on emploie P^f+ au lieu de P, ou P&LC^f+ au lieu de P&LC, ou P&LC&C^f+ au lieu de P&LC&C. Le nombre d'états mémorisés est divisé par 3,3 quand on emploie P^f+ au lieu de P, par près de 3 quand on emploie P&LC^f+ au lieu de P&LC et par près de 3 encore quand on emploie P&LC&C^f+ au lieu de P&LC&C. Malgré le supplément de travail correspondant aux états créés sans évaluation, le temps est divisé par près de 10 quand on passe de P à P^f+, divisé par 2,3 quand on passe de P&LC à P&LC^f+ et dinminué de 36 % quand on passe de P&LC&C à P&LC&C^f+.

Dans le tableau 5, les nombres d'états visités baissent sensiblement au fur et à mesure que l'heuristique est plus sophistiquée.

La baisse en temps apportée par notre technique est de 29 % lorsqu'on l'applique à P seule, 45 % lorsqu'on l'applique à P&LC, 18 % lorsqu'on l'applique à P&LC&C.

Cependant, le temps de calcul pour LC et C ralentit manifestement le traitement, (par exemple, P&LC utilise 50 % de temps supplémentaire par rapport à P). Les heuristiques LC et C, conçues par Korf pour attaquer le taquin 5 x 5, ne sont pas pleinement déployées pour le taquin 3 x 3 (il faut alors tenir compte d'interférences entre elles).

	développements	états mémorisés	états créés sans évaluation	temps
P	1 045	1 612	0	1*
P^f	320	508	2 235	0,117
P^{f+}	310	487	1 611	0,105
P&LC	517	813	0	0,239
P&LC^f	184	297	2 830	0,104
P&LC^{f+}	173	274	1 986	0,100
P&LC&C	319	513	0	0,114
P&LC&C^f	122	202	3 453	0,094
P&LC&C^{f+}	109	176	2 431	0,073

Tableau 4. Résultats moyens obtenus en appliquant A* à 200 problèmes 3 x 3 tirés au sort. Le temps d'exécution avec P sert de référence (1* dénote environ 12,25 s sur notre Mac G4)

	états visités	passes	créés sans évalu.	temps
P	3 454	5,1	0	1*
P^f	980	4,1	7 048	0,84
P^{f+}	964	4,1	5 672	0,71
P&LC	1 627	4,5	0	1,56
P&LC^f	455	3,5	6 844	1,11
P&LC^{f+}	439	3,5	5 371	0,85
P&LC&C	975	4,1	0	0,97
P&LC&C^f	277	3,1	7 160	0,90
P&LC&C^{f+}	255	3,1	5 520	0,80

Tableau 5. Résultats moyens obtenus en appliquant IDA* aux mêmes 200 problèmes 3 x 3. Le temps d'exécution avec P sert de référence (1* représente environ 2 s sur notre Mac G4).

Dans le tableau 6, les nombres d'états visités baissent sensiblement aussi au fur et à mesure que l'heuristique est plus sophistiquée. La baisse en temps apportée par notre technique est de 52 % lorsqu'on l'applique à P, de 63 % lorsqu'on l'applique à P&LC, 38 % lorsqu'on l'applique à P&LC&C.

	états visités	passes	états créés sans évaluation	temps
P	1 117 438	6,8	0	1*
P^f	182 813	5,8	1 830 955	0,56
P^{f+}	182 785	5,8	1 518 489	0,48
P&LC	223 537	6,1	0	0,78
P&LC^f	34 921	5,1	1 025 491	0,34
P&LC^{f+}	33 625	5,1	786 271	0,29
P&LC&C	105 423	5,8	0	0,39
P&LC&C^f	16 254	4,8	838 352	0,27
P&LC&C^{f+}	16 220	4,8	687 141	0,24

Tableau 6. Résultats moyens obtenus en appliquant IDA* aux problèmes $K_{4,1}$ à $K_{4,10}$. Le temps d'exécution avec P sert de référence (1* dénote environ 16 mn 10 s sur notre Mac G4)

Quand on applique A* au taquin 4 x 4, les temps de calculs deviennent lourds voire exorbitants. Nous n'avons pas mené à bien, sur les exemples 4 x 4 de Korf, les essais avec A* muni successivement de chacune des 9 heuristiques du tableau 4 : la tâche était impraticable (vu nos ressources informatiques ici) en employant les heuristiques les plus faibles.

Cependant, nous avons comparé P&LC&C et P&LC&C^{f+}, avec A*, sur les problèmes $K_{4,1}$ à $K_{4,20}$:

	développements	états mémorisés	états créés sans évaluation	temps
P&LC&C	57 971	108 832	0	8 h 26 mn
P&LC&C^{f+}	11 655	22 034	672 290	28 mn 31 s

Tableau 7. Résultats moyens obtenus en appliquant A* aux problèmes $K_{4,1}$ à $K_{4,20}$.
Le temps d'exécution est mesuré sur notre Mac G4.

On constate qu'avec notre technique le nombre de développements et le nombre d'états mémorisés sont divisés par près de 5. Et, malgré une moyenne de 672 290 "états créés sans évaluation", le temps est divisé par près de 18.

Notons que la moyenne des états mémorisés (22 034 seulement avec P&LC&C^{f+} pour les 20 premiers problèmes de Korf) est très en-dessous de la barrière de 30 000 états mémorisés simultanément que Korf n'avait pu franchir en 1985 pour aucun de ses 100 problèmes[15].

Et nous terminons par la comparaison entre les performances de P&LC&C et P&LC&C^{f+}, lorsqu'appliqués tous deux avec IDA*, aux exemples $K_{4,1}$ à $K_{4,50}$ de Korf. Le gain en temps est de 38 %. Nous présumons que ce gain peut être accru par une programmation plus soignée que celle que nous avons réalisée pour l'instant.

	états visités	passes	états créés sans évaluation	temps
P&LC&C	1 212 433	6,4	0	1*
P&LC&C^{f+}	191 258	5,6	10 038 543	0,62

Tableau 8. Résultats moyens obtenus en appliquant IDA* aux problèmes $K_{4,1}$ à $K_{4,50}$. Le temps de P&LC&C sert de référence (1* représente ≅ 1 h 17 mn 36 s sur notre Mac G4)

8 Remarques de conclusions

L'idée principale, originale à notre connaissance, qui a donné lieu à ce travail est que les heuristiques sont sous-employées quand on les résume à des fonctions numériques, que ce faisant on oublie les hypothèses sous-jacentes à leur

[15] Avec P il est vrai. Mais même avec P&LC&C la barrière des 30 000 états mémorisés simultanément est dépassée par la plupart des 20 problèmes $K_{4,1}$ à $K_{4,20}$ (sauf $K_{4,2}$ à $K_{4,4}$, qui exigent tous 3 une moyenne proche de 27 000 états mémorisés). Par exemple : 35 128 états mémorisés pour $K_{4,1}$, 146 653 pour $K_{4,5}$, 248 227 pour $K_{4,7}$.

constitution et que cette perte d'information est dommageable. A l'inverse, nous avons proposé de tirer parti de la "raison" des heuristiques, pas seulement de leur "valeur" numérique réductrice.

Nous avons testé cette idée sur les taquins 3 x 3 et sur des jeux de taquins 4 x 4 de référence (problèmes de Korf).

Les premiers résultats sont très encourageants ; pour exécuter A* nous sommes sensiblement moins limités en mémoire qu'avec le mode classique d'exploitation des heuristiques ; pour exécuter A* ou IDA* nous gagnons très significativement en temps.

Avoir travaillé avec IDA* et A* n'est a priori pas restrictif : l'approche proposée en vue de mieux tirer parti des heuristiques est indépendante de l'algorithme de recherche considéré.

On peut s'attendre à d'autres améliorations avec des algorithmes qui concurrencent A* ou IDA*. Mais ceci reste à vérifier.

Nous souhaitons également expérimenter l'approche avec d'autres heuristiques concernant les taquins et pour d'autres problèmes que les taquins.

Notre implémentation (des algorithmes et des heuristiques) peut sans doute être améliorée mais les éventuels perfectionnements retentiraient certainement dans des proportions semblables d'une part sur la résolution classique et d'autre part sur la résolution selon notre approche. Pour les taquins, cette approche se prête particulièrement à la parallélisation.

Nous comptons approfondir les formalisations que nous avons engagées ici autour du concept de h-résolvabilité. Une modélisation prenant en compte les poids respectifs des tests de h-résolvabilité et de la résolution classique est souhaitable.

9 Références

[1] A. Brungger,, Solving hard combinatorial optimization problems in parallel : two case studies, Ph. D. Thesis, ETH Zürich, 1997.

[2] A. Brungger, A. Marzetta, K. Fukuda, J. Nievergelt J., The parallel search bench ZRAM ant its applications, *Annals of Operations Research*, Vol. 90, pp 45-63, 1999.

[3] G. Cousineau, M. Maury, *Approche fonctionnelle de la programmation*, Édisciences, 1995.

[4] J. Culberson, J. Schaeffer, Pattern Databases, *Computational Intelligence*, Vol. 14, N° 4, pp 318-334, 1998.

[5] H Farreny, M. Ghallab *Éléments d'Intelligence Artificielle*. Hermès, 1987.

[6] H. Farreny, *Recherche Heuristiquement Ordonnée, Algorithmes et propriétés*, Masson, 1995.

[7] H. Farreny, Recherche Heuristiquement Ordonnée : généralisations compatibles avec la complétude et l'admissibilité, *Technique et Science Informatiques*, Vol. 16, N° 7, pp 925-953, 1997.

[8] H. Farreny, Completeness and admissibility for general heuristic search algorithms. A theoretical study : basic concepts and proofs, *Journal of Heuristics*, Vol. 5, pp 353-376, 1999

[9] M. Gardner, *Jeux mathématiques du "Scientific American"*, CEDIC, 1979

[10] R. Gasser, Harnessing computational resources for efficient exhaustive search, Ph. D. Thesis, Swiss Federal Institute of Technology, Zürich, Switzerland, 1995

[11] O. Hansson, A. Mayer, M. Yung, Criticizing solutions to relaxed models yields powerful admissible heuristics, *Information Sciences*, Vol. 63, N° 3, pp 207-227, 1992.

[12] P. E. Hart, N. J. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths, *IEEE Transactions on Systems Science and Cybernetics*, Vol SSC-4, N° 2, 1968.

[13] W. W. Johnson, W. E. Storey, Notes on the '15' puzzle, *American Journal of Mathematics*, Vol. 2, pp 397-404, 1879.

[14] L. Kanal, V. Kumar (Editors), *Search in Artificial Intelligence*, Springer-Verlag, 1988.

[15] R. E. Korf, Depth-first iterative-deepening : An optimal admissible tree search, *Artificial Intelligence*, Vol. 27, N° 1, pp 97-109, 1985.

[16] R. E. Korf, Real-time heuristic search, *Artificial Intelligence*, Vol. 42, pp 189-211, 1990.

[17] R. E. Korf, Linear-space best-first search, *Artificial Intelligence*, Vol. 62, pp 41-78, 1993.

[18] R. E. Korf, L. A. Taylor, Finding optimal solutions to the twenty-four puzzle, *Proceedings of AAAI-96*, pp 1202-1207, Portland, 1996.

[19] R. E. Korf, Recents progress in the design and analysis of admissible heuristic functions, *Proceedings of AAAI-00*, pp 1165-1170, Austin, 2000.

[20] D. Kornhauser, G. Miller, P. Spirakis, Coordinating pebble motion on graphs, the diameter of permutation groups, and applications, *25th FOCS*, pp. 241-250, 1984

[21] N. J. Nilsson, *Problem-solving methods in Artificial Intelligence*, Mc Graw Hill, 1971.

[22] N. J. Nilsson, *Principles of Artificial Intelligence*, Morgan Kaufmann, 1980. *Principes d'Intelligence Artificielle*, Cepadues, 1988.

[23] N. J. Nilsson, *Artificial intelligence : a new synthesis*, Morgan Kaufmann, 1998.

[24] J. Pearl, *Heuristics : intelligent search strategies for computer problem-solving*, Addison-Wesley, 1984.

[25] A. E. Prieditis, Machine discovery of effective admissible heuristics, *Machine Learning*, Vol. 12, pp 117-141, 1993.

[26] D. Ratner, M. Warmuth, The (n^2-1)-puzzle and related relocation problems, *Journal of symbolic computation*, Vol. 10, pp 111-137, 1990.

[27] A. Reinefeld, Complete solution of the eight-puzzle and the benefit of node ordering in IDA*, *Proc. 13th IJCAI*, Chambéry, pp 248-253, 1993.

[28] A. Reinefeld, Enhanced iterative-deepening search, I*EEE transactions on pattern-analysis and machine intelligence*, Vol. 16, N° 7, pp 701-710, 1994.

[29] P. D. A. Schofield, Complete solution of the 'Eight-Puzzle', *Machine Intelligence 1*, pp 125-133, Edinburgh University Press, (réédition en) 1971 (1e éd. : 1967).

[30] B. S. Stewart, C. F. Liaw, C. C. White III, A bibliography of heuristic search through 1992, *IEEE Trans. on Systems, Man and Cybernetics*, Vol. 24 , N° 2, pp 268-293, 1994.

[31] L. A. Taylor, R. E. Korf, Pruning duplicate nodes in depth-first search, *Proc. 11th AAAI*, Washington DC, pp 756-761, 1993.
[32] R. M. Wilson, Graph puzzles, homotopy and the alternating group, *Journal of Combinatorial Theory*, Vol. B, N° 16, pp 86-96, 1974.