Quand la TI 58 résout un problème d'échecs

La TI 58 ne gagnera sans doute jamais une partie d'échecs : sa mémoire est trop limitée pour contenir un programme même mauvais joueur. En revanche, elle peut résoudre certains problèmes. En voici la preuve.

■ Les échecs sont considérés depuis très longtemps comme étant le jeu intellectuel par excellence. C'est logique car excepté le go, qui est beaucoup moins pratiqué en Occident, le jeu d'échecs possède l'arbre (de jeu) le plus important. Il est donc normal qu'il ait été le premier à intéresser les informaticiens et que l'écriture d'un bon programme d'échecs soit devenue synonyme de création d'une intelligence artificielle.

Evidemment, il ne faut pas rêver : on ne peut pas espérer réaliser un programme d'échecs même très moyen sur un op actuel. Néanmoins, on peut y résoudre des problèmes plus simples se rapportant aux échecs tels que le problème du placement sans prise réciproque de huit reines sur un échiquier.

Ce problème n'est pas nouveau mais il n'avait à ma connaissance jamais été traité sur un op, et de plus la technique employée ici est intéressante puisqu'elle est utilisée dans la plupart des programmes de jeu : il s'agit de l'exploration d'arborescence très chère à David Levy (1). Nous verrons notamment com-

(1) Voir Les jeux et l'ordinateur *de David Levy, traduit en français par Jean-Pierre Brunerie, et publié dans les n*^{os} *16 à 35 de* L'Ordinateur Individuel.

Quand la TI 58 résout un problème d'échecs

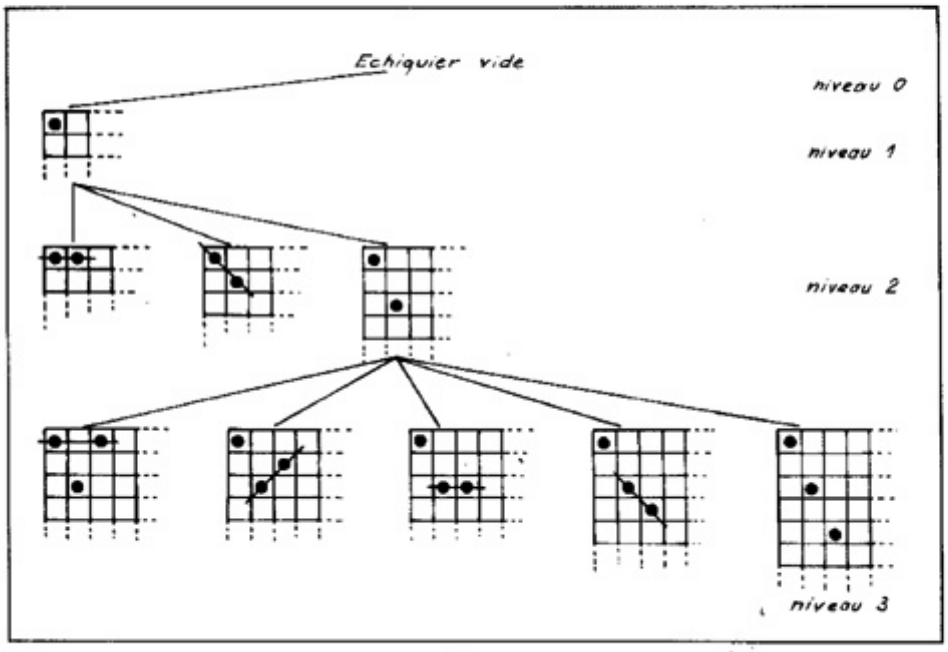


Fig. 1
Si l'arbre est immense, il compte beaucoup de branches qu'il n'est pas utile d'explorer.

ment examiner un arbre assez touffu en ne gardant en mémoire qu'un minimum de positions.

Rappelons l'énoncé du problème : il s'agit de placer sur un échiquier (8x8) un nombre maximum de reines de telle manière qu'aucune ne puisse en prendre une autre. A priori il y a $\sum_{i=0}^{64} C_{64}^{i} = 2^{64} > 10^{19}$ manières de placer un certain nombre de reines sur un échiquier, mais comme une reine peut prendre toute reine située sur la même verticale, on aura au maximum une reine par colonne et donc on ne peut espérer placer plus de 8 reines au total.

Mais il y a encore 8^8 = 16 777 216 manières de placer ces 8 reines et si l'on estime que l'on évalue une position en une seconde (ce qui paraît raisonnable sur un op), il faudra plus de six mois pour les examiner toutes. Heureusement on va voir que l'on peut très vite élaguer l'arbre et éliminer de nombreuses positions. Néanmoins l'exécution du programme sera encore relativement longue.

Le sommet de l'arbre est constitué par l'échiquier vide (position de départ). On commence par descendre dans l'arbre en plaçant une reine en haut de la 1re colonne (fig. 1), on continue ensuite à descendre en cherchant à placer une reine dans la 2e colonne. Sur la 1re ligne c'est évidemment impossible car on aurait deux reines sur une même horizontale ; il est donc inutile d'examiner l'arbre plus profondément à partir de cette position en cherchant à placer des reines dans les colonnes suivantes. On élimine ainsi 8^6, soit 262 144 autres positions. On examine alors la position suivante au même niveau en plaçant la reine de la 2e colonne sur la 2e ligne. Mais c'est également impossible : 2 reines sur une même diagonale... donc même scénario et de nouveau 8^6 positions éliminées. La position suivante au même niveau est obtenue en plaçant cette reine sur la 3e ligne où elle ne peut plus être prise.

On continue alors à descendre dans l'arbre en cherchant à placer une reine dans la 3e colonne. Sur la 1re ligne, elle est prise par la 1re reine ; 2e, 3e et 4e ligne, elle est prise par la 2e reine, ce qui fait encore $4x8^5$, soit 131 072 positions éliminées et elle se place finalement à la 5e ligne.

On continue toujours à descendre ainsi dans l'arbre en cherchant à placer des reines dans les colonnes suivantes jusqu'à ce que l'on trouve une colonne où c'est impossible : toutes les lignes sont en prises (fig. 2). On remonte alors d'un niveau dans l'arbre (colonne précédente), et l'on examine la position suivante sans prise à ce niveau (si c'est impossible, on remonte encore d'un niveau et ainsi de suite) et puis l'on redescend dans l'arbre (colonne

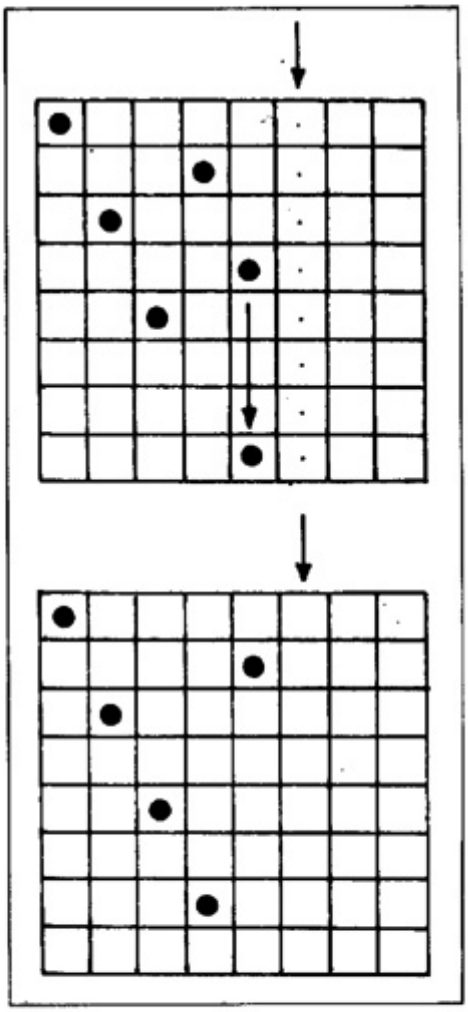
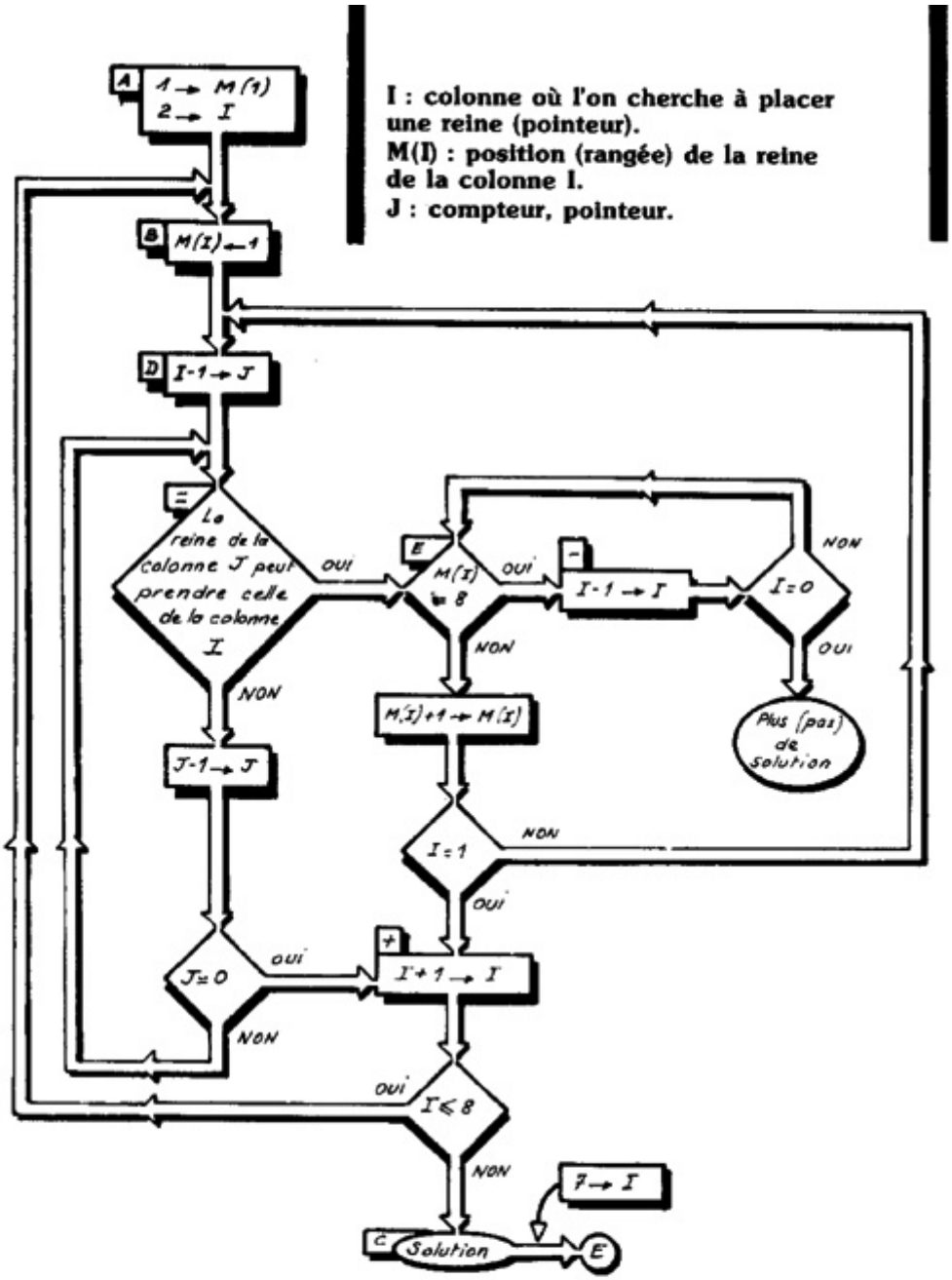


Fig. 2
Dans la colonne 6, il est impossible de placer une reine qui ne soit prise par une autre. On remonte donc d'un niveau dans l'arbre. Revenu dans la colonne 5, on examine la position suivante sans prise à ce niveau, c'est la ligne 8. S'il n'y avait pas eu de solution, on serait encore remonté d'un niveau : colonne 4, et ainsi de suite. Quand on a trouvé une position sans prise, on redescend dans l'arbre : colonne 6 (ou 5, si l'on est remonté jusqu'à la colonne 4) et l'on cherche à placer une reine dans cette colonne en commençant par la ligne n° 1.



I : colonne où l'on cherche à placer une reine (pointeur).
M(I) : position (rangée) de la reine de la colonne I.
J : compteur, pointeur.

suivante). Le programme s'arrête si une solution est trouvée (les 8 reines sont correctement placées) ou si l'on est remonté jusqu'au sommet de l'arbre c'est-à-dire s'il n'y a plus (ou pas) de solutions (affichage "0" clignotant). Toutes les positions ont alors été examinées.

On trouvera cet algorithme décrit dans l'organigramme ci-dessus.

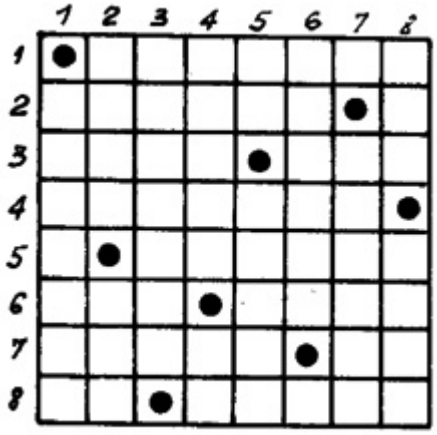
Le programme n'est pas très long (119 pas) et il n'utilise que 10 mémoires :

00 : colonne de la reine que l'on cherche à placer		(I)
01 : position (ligne) de la reine de la colonne	1	M
08 :	8	
09 : compteur et pointeur (indice)		(J)

On considère donc un vecteur M(i) à 8 composantes rangées dans les mémoires 01 à 08 ; les mémoires 00 et 09 servant d'indices.

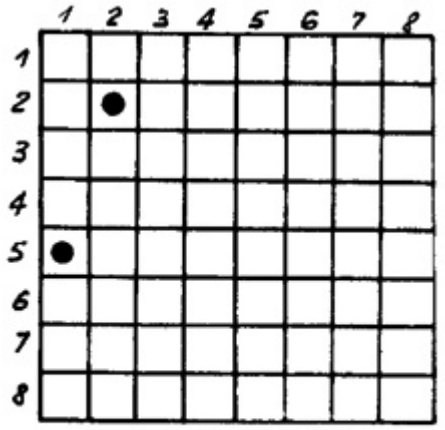
Pour exécuter ce programme, on appuie simplement sur CLR et A et, après 53 minutes, on voit apparaître enfin 15863724. Le 1er chiffre est le contenu de la mémoire 01 et ainsi de suite jusqu'au 8e chiffre ; ce résultat est obtenu par une petite astuce qui se trouve sous le label C dans le programme.

On peut alors représenter la situation sur l'échiquier et vérifier que c'est bien une solution. Il suffit ensuite d'appuyer successivement sur R/S pour obtenir les solutions suivantes. On obtient de cette manière en ± 16 heures 96 positions différentes. Attention toutefois : certaines d'entre elles sont semblables à un isomorphisme près, c-à-d. que l'on peut les appliquer l'une sur l'autre par une rotation ou une symétrie, mais ces notions de géométrie analytique sortent du cadre de ce programme.



Pour les personnes pressées, le programme permet aussi de démarrer l'exploration à un endroit quelconque de l'arbre. On peut ainsi espérer obtenir des solutions plus rapidement.

Supposons par exemple que l'on veuille démarrer l'exploration à partir de la position suivante :



On entre d'abord 5 STO 01 et 2 STO 02 puis 3 STO 00 (n° de la colonne où démarre l'exploration) et l'on presse enfin B. Après 5 mn environ, on obtient 52468317. Il faut dans ce cas bien veiller à ce que la position partielle que l'on entre soit sans prise réciproque.

Il est possible d'obtenir une solution encore plus rapidement en entrant simplement 5 STO 01 2 STO 00 B. Après 2 mn 28 sec, on obtient 51468273. On peut de la même manière commencer l'exploration à partir d'une solution déjà trouvée (par programme ou d'une autre manière). Il suffit de rentrer la position dans les mémoires 1 à 8 et d'appuyer sur C. Après environ 8

Quand la TI 58 résout un problème d'échecs

secondes, on visualise la position à l'écran et on appuie ensuite sur R/S pour obtenir une autre solution. Pour voir ou revoir la situation à un moment quelconque entrer CLR et presser sur C.

Une première généralisation consiste à chercher des solutions pour un "échiquier" nxn avec n ≠ 8. Les modifications à apporter au programme sont minimes ; il suffit de remplacer le 8 des pas 104, 093, 067 et 050 par la dimension n souhaitée. On peut faire encore mieux en stockant cette dimension dans une mémoire non utilisée (10 ou autres) au début de l'exécution du programme... 2nd Lbl A STO 10 1 STO 01... et en la rappelant aux pas indiqués ci-dessus (RCL 10).

Le problème des reines.
Programme pour TI 58/59
Auteur Alain Daix

000	76	LBL	039	95	=	078	00	00
001	12	B	040	67	EQ	079	67	EQ
002	01	1	041	15	E	080	85	+
003	72	ST*	042	97	DSZ	081	14	D
004	00	00	043	09	09	082	76	LBL
005	43	RCL	044	95	=	083	11	A
006	00	00	045	76	LBL	084	01	1
007	76	LBL	046	85	+	085	42	STO
008	14	D	047	43	RCL	086	01	01
009	42	STO	048	00	00	087	02	2
010	09	09	049	32	X↕T	088	42	STO
011	69	OP	050	08	8	089	00	00
012	39	39	051	67	EQ	090	12	B
013	73	RC*	052	13	C	091	76	LBL
014	00	00	053	69	OP	092	13	C
015	32	X↕T	054	20	20	093	08	8
016	76	LBL	055	12	B	094	42	STO
017	95	=	056	76	LBL	095	09	09
018	73	RC*	057	75	-	096	25	CLR
019	09	09	058	69	OP	097	76	LBL
020	67	EQ	059	30	30	098	43	RCL
021	15	E	060	29	CP	099	85	+
022	85	+	061	43	RCL	100	73	RC*
023	43	RCL	062	00	00	101	09	09
024	00	00	063	67	EQ	102	65	×
025	75	-	064	91	R/S	103	53	(
026	43	RCL	065	76	LBL	104	08	8
027	09	09	066	15	E	105	75	-
028	95	=	067	08	8	106	43	RCL
029	67	EQ	068	32	X↕T	107	09	09
030	15	E	069	73	RC*	108	54	)
031	73	RC*	070	00	00	109	22	INV
032	09	09	071	67	EQ	110	28	LOG
033	85	+	072	75	-	111	97	DSZ
034	43	RCL	073	01	1	112	09	09
035	09	09	074	74	SM*	113	43	RCL
036	75	-	075	00	00	114	95	=
037	43	RCL	076	32	X↕T	115	91	R/S
038	00	00	077	43	RCL	116	69	OP
						117	30	30
						118	15	E
						119	00	0
						120	00	0

Pour le démarrage, on entrera alors n puis on appuiera sur A. Le programme compte maintenant 125 pas.

Attention : Si n ⩾ 9 la mémoire 09 contiendra la position de la "reine" de la colonne 9 et donc il faudra utiliser une autre mémoire comme indice de vecteur. Avec le programme modifié, on trouve qu'il n'y a pas de solution pour n = 2 en 11 secondes et pour n = 3 en 35 secondes. Et pour n = 5, on obtient 13524 en 42 secondes.

Tout ce qui a été dit pour un échiquier 8x8 reste valable ici et notamment la possibilité de démarrer l'exploration à un endroit quelconque de l'arbre. De plus il est possible de chercher une solution pour un n donné à partir d'une solution pour un n plus petit et cela de la même manière.

Une autre généralisation beaucoup plus intéressante, mais aussi nettement moins évidente à mettre en œuvre, trouve son origine dans un article de la revue *Amusements in Math* publié en 1917. Le problème proposé par Dudeney et portant le nom de "The no-three-in-line-problem" était le suivant : "Dans un réseau nxn, placer un nombre maximum de points de telle manière que la droite passant par 2 points quelconques n'en contienne pas un troisième."Il y a quatre ans, le problème n'était toujours pas résolu pour n > 16 mais je ne pense pas qu'il soit possible d'aller beaucoup plus loin grâce à l'ordinateur car les temps d'exécution sont énormes. Néanmoins j'ai écrit un programme pour TI 58 qui pour des n petits donne toutes les solutions dans des temps raisonnables.

On peut aller encore plus loin en considérant un réseau à m dimensions nxnxn... xn et en cherchant à placer un nombre maximum de points de telle manière que l'espace à m-1 dimensions passant par m-1 points quelconques n'en contienne pas un mième. Mais là je dois vous avouer que, même à 3 dimensions pour un réseau nxnxn et un nombre maximum de points tels que le plan passant par trois points quelconques n'en contienne pas un quatrième, je n'ai pas encore envisagé le problème.

Si un lecteur a des idées sur le sujet, je serais heureux qu'il m'en fasse part.

☐ Alain Daix