# Intégration de l'apprentissage non supervisé aux métaheuristiques pour la résolution des problèmes d'optimisation combinatoire difficiles

Mahmoud Zennaki[1], Ahmed Ech-cherif[1]

[1] Département Informatique, Faculté des Sciences, U.S.T.O.M.B.
BP 1505 El-M'naouer – ORAN
{mzennaki, a-echerif}@univ-usto.dz

**Résumé.** Le présent article rentre dans le cadre d'un projet ambitieux qui a pour objectif l'intégration des techniques d'apprentissage aux métaheuristiques pour contribuer à la résolution des problèmes d'optimisation combinatoire NP-difficiles. L'approche proposée consiste à incorporer un apprentissage non supervisé aux métaheuristiques à population de solutions pour « apprendre » et constituer des clusters de solutions de caractéristiques différentes à partir de l'histoire de recherche. Cette forme de connaissance extraite sera utilisée pour guider la métaheuristique vers des régions de solutions prometteuses. L'algorithme proposé, et dérivé de cette approche, est une extension de la recherche par dispersion et peut automatiquement apprendre pendant le processus de recherche en exploitant l'histoire de recherche représenté par l'ensemble des solutions trouvées. Les résultats préliminaires avec une application au problème très connu des tournées de véhicules (VRP) montrent le grand intérêt d'une telle approche qui peut effectivement trouver des solutions de haute qualité pour des instances de grande taille d'un problème donné.

**Mots clés :** Apprentissage machine, Apprentissage non supervisé, Clustering, Recherche réactive, Séparateurs à vaste marge, Recherche par dispersion, Construction de vocabulaire.

## 1 Introduction

Les heuristiques restent le seul moyen pour traiter des instances de grande taille de problèmes d'optimisation combinatoire et pour obtenir des solutions proches de l'optimum en un temps raisonnable. Cependant le succès de telles méthodes dépendent largement d'un paramétrage adéquat de certains paramètres tels que le degré de récence dans la recherche tabou, la température initiale et finale dans le recuit simulé etc. Le processus de paramétrage est généralement géré manuellement à travers des simulations sur un ensemble d'instances. Plusieurs tentatives ont été faites pour automatiser le processus de paramétrage sur un certain nombre de métaheuristiques classiques lors de la résolution d'instances spécifiques de problèmes d'optimisation, et des résultats encourageants ont été obtenus sur une large variété de

problèmes allant du problème de la clique maximale, problème du voyageur de commerce, problème d'arrangement linéaire, jusqu'au problème de l'exploration de la planète Mars par des véhicules coopératifs [1, 2, 3, 4, 5, 19, 20, 22, 24].

Dans [5], les auteurs ont introduit une approche générale proposant l'utilisation d'algorithmes d'apprentissage pour apprendre et mémoriser différentes régions de l'espace de recherche. Dans [2], les auteurs ont adapté ces techniques à la recherche tabou en utilisant un degré de récence auto-paramètrable qui permet une oscillation adéquate entre les phases d'intensification (exploration plus intensive d'une petite région prometteuse de solutions) et de diversification (recherche de meilleures solutions dans d'autres régions de l'espace de recherche) ainsi qu'une utilisation efficace de l'histoire de recherche par l'implémentation de structures de données et méthodes d'accès performantes tels que le hachage combiné aux arbres binaires.

Notre objectif est d'introduire un paradigme général en intégrant des méthodes de classification non supervisée à des métaheuristiques à populations de solutions en développant un algorithme de recherche réactive qui peut apprendre pendant le processus de recherche. Notre approche peut être structurée comme suit.

Génération d'une population de solutions ; une heuristique de recherche locale basée sur une structure de voisinage adaptée au problème d'optimisation combinatoire à résoudre, est utilisée pour générer une population de solutions.

Construction du vocabulaire ; les attributs ou caractéristiques des solutions obtenues sont utilisés pour construire un vocabulaire de la même manière que dans la recherche par dispersion [12].

Classification automatique par clustering ; un clustering sous forme de séparateurs à vaste marge de type non supervisé [23] est accompli sur le vocabulaire préalablement construit afin de déterminer des sous ensembles de solutions similaires qui permettront en retour de générer des solutions plus représentatives.

Génération de solutions élites ; les clusters obtenus dans l'étape précédente à l'aide d'une classification automatique sont alors utilisés pour générer des solutions élites. Cette combinaison de solutions peut aussi être vue comme une diversification du processus de recherche en recherchant de meilleures solutions dans des régions non encore explorées.

Le reste de l'article est organisé comme suit. Dans la section 2, nous passons en revue la technique de clustering utilisée dans notre algorithme basée sur les séparateurs à vaste marge de type non supervisé. La section 3 décrit la génération de solutions élites obtenues via une classification automatique de la population de solutions. Dans la section 4, une application de notre algorithme au fameux problème des tournées de véhicules (VRP) est présentée ainsi que des résultats d'expérimentation. Une conclusion est donnée à la fin de l'article, accompagnée de remarques intéressantes.

## 2 Classification par USVM

Les SVMs [23] ont été développés par V.Vapnik et al. basés sur la théorie d'apprentissage statistique. Dans leur forme de base, les SVMs sont utilisés dans l'apprentissage supervisé, où les étiquettes des exemples (classes de solutions dans

notre cas) sont connues. Etant donné un ensemble de n exemples d'apprentissage $S_n=((x_1,y_1),\ldots,(x_2,y_2))$, $x_i \in IR^n$ représentent les exemples, $y_i \in \{-1,+1\}$ les étiquettes. Les SVMs permettent de trouver une règle de décision sous la forme d'un hyperplan séparant les exemples d'apprentissage avec la plus grande distance euclidienne possible. Cette distance est appelée marge δ, comme décrit dans la figure 1.

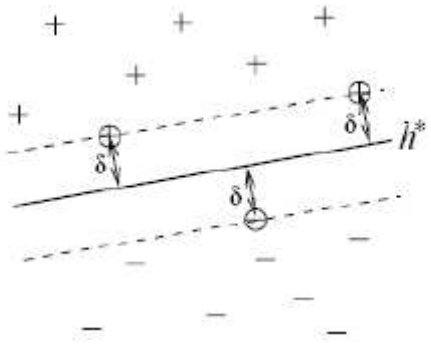


**Fig. 1.** Le plan optimal h* est celui qui maximise la marge δ

Pour permettre une séparation linéaire, une fonction noyau est utilisée pour projeter l'ensemble initial des exemples dans un espace de grande dimension. La fonction noyau représente la répartition des données dans cet espace. Cette notion confère aux SVMs une flexibilité remarquable, un temps d'exécution raisonnable et une capacité de généralisation même en présence d'un nombre important d'attributs.

Mais dans notre cas, les classes de solutions sont inconnues. Nous sommes donc dans le cas de la classification non supervisée ou plus simplement clustering. Nous utilisons pour cela les SVMs de type non supervisé appelés USVM (Unsupervised SVM), qui représentent une variante des SVMs qui manipulent seulement des exemples non étiquetés. Au lieu de trouver un hyperplan qui maximise la marge, USVM consiste à trouver les étiquettes des exemples de manière à les séparer avec la plus grande marge possible (voir figure 2). Pour un état de l'art récent sur les USVM, le lecteur est référé à [9, 21].

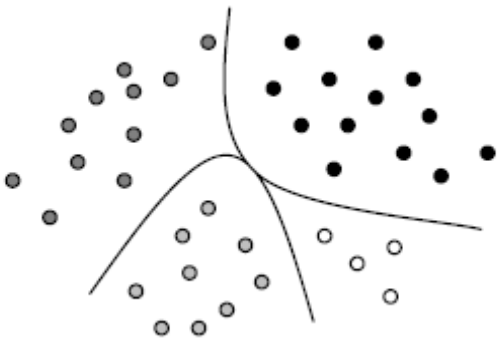

**Fig. 2.** Unsupervised SVM

# 3 Génération des solutions élites

## 3.1 Etat de l'art

La notion de population de solutions utilisée dans certaines métaheuristiques représente la forme la plus courante de représentation de l'histoire de recherche comme dans la recherche tabou ou les algorithmes génétiques. En effet, nous pouvons considérer qu'une population de solutions est une sorte de mémoire qui représente l'évolution du processus de recherche. La différence principale entre ces algorithmes provient dans la manière de construction d'une telle population, et comment de nouvelles solutions sont construites. Plusieurs algorithmes ont été proposés notamment dans [10, 12, 13] pour la construction de populations de solutions. Une idée consiste à construire un certain nombre de nouvelles solutions avec la population de solutions courante. Ces nouvelles solutions formeront la population de la nouvelle génération, ce qui signifie que l'ancienne population est complètement remplacée. Au lieu du remplacement, il est aussi possible de remplacer seulement une partie de cette population. L'autre extrême, est de remplacer une solution par une autre de meilleure qualité. Nous avons dans la littérature plusieurs stratégies telles que l'élimination des anciennes solutions, l'élimination aléatoire ou aussi l'élimination des solutions de mauvaises qualités. Ces stratégies affectent de manière considérable la vitesse de convergence de telles métaheuristiques. Nous pouvons voir par exemple que le remplacement de mauvaises solutions conduit à une meilleure convergence comparée aux autres stratégies. Néanmoins, cette stratégie conduit généralement à une population de solutions très similaires. Pour améliorer la qualité des populations générées, nous pouvons augmenter par exemple la taille de la population. Plus de détails théoriques sur la convergence de tels algorithmes sont présentés dans [15].

Dans la recherche par dispersion, Glover [12] suggère d'éviter la convergence de la population avec une politique plus élaborée d'élimination de solutions. La population conserve non seulement les meilleures solutions mais aussi certaines solutions spéciales caractérisées par des attributs de types différents. C'est cette stratégie qui a été adopté dans notre algorithme pour la partie génération de population. Ceci se justifie par le fait que nous voulons réaliser un apprentissage automatique sur un ensemble de solutions ayant le maximum d'attributs différents.

## 3.2 Génération des populations et construction du vocabulaire

La génération de populations de solutions nécessite l'utilisation de méthodes heuristiques de recherche locale basée sur une structure de voisinage adaptée au problème d'optimisation considérée. Au lieu de rechercher des optimums locaux comme dans certaines heuristiques classiques, ou générer un ensemble de solutions pour ne retenir que les meilleurs comme dans la recherche tabou [11, 16], notre méthode essaye de générer un nombre important de bonnes mais aussi de mauvaises solutions. Dans le cas où le problème contient un nombre important de contraintes qui réduisent la taille de l'espace des solutions réalisables, une relaxation est faite sur

certaines contraintes en ajoutant des pénalités à la fonction économique, afin d'augmenter le nombre de solutions générées.

Les attributs des solutions ainsi obtenues sont utilisés pour construire un vocabulaire. Ce terme de vocabulaire a été déjà utilisé dans la recherche par dispersion [11, 12]. Par exemple, dans le cas du problème des tournées de véhicules (VRP), une solution est un ensemble de tournées qui satisfont certaines contraintes et le vocabulaire peut être considéré comme l'ensemble des tournées réalisables trouvées dans la population. Ces tournées seront par la suite combinées pour générer de nouvelles solutions. Pour palier au problème du stockage de l'histoire de recherche, nous utilisons des structures de données et méthodes d'accès spécifiques comme décrites dans [2].

### 3.3 Classification automatique

Au lieu d'utiliser un opérateur de croisement comme dans les algorithmes génétiques [13, 15], nous réalisons une classification automatique de la population basée sur le vocabulaire construit précédemment. Les classes de solutions ainsi obtenues sont utilisées pour générer des solutions élites souvent de haute qualité. Cette génération pourra se faire soit en sélectionnant une solution par classe pour former la nouvelle génération, soit en générant de nouvelles solutions construites à partir d'attributs des solutions appartenant aux différentes classes obtenues par la classification.

## 4 Application au problème des tournées de véhicules

Le problème des tournées de véhicule (Vehicle Routing Problem VRP) est l'un des problèmes d'optimisation combinatoire les plus difficiles et les plus étudiées aussi. Défini il y a déjà 40 ans, ce problème consiste en la recherche d'un ensemble optimal de tournées pour une flotte de véhicules dans le but de servir un ensemble donné de clients. L'intérêt que portent les chercheurs à ce problème provient de sa difficulté mais aussi ses domaines d'application très variés. Le problème VRP est un problème de programmation en nombre entiers donc NP-complet, ce qui signifie que l'effort requis pour résoudre ce problème augmente de manière exponentielle en fonction de la taille du problème. Pour de tels problèmes, l'utilisation d'heuristiques efficaces est fortement recommandée surtout pour des instances de grande taille.

Le problème des tournées des véhicules apparaît lorsqu'un ensemble de tournées pour une flotte de véhicules basés à un ou plusieurs dépôts doivent être déterminées pour un certain nombre de clients dispersés géographiquement. L'objectif du VRP est de servir la demande de ces clients avec un coût minimal en partant puis en revenant aux dépôts. Les notations utilisées pour la formulation du VRP sont basées sur un graphe simple G(V,U) avec :

$V = \{v_0, v_1, \ldots, v_n, v_{n+1}\}$ un ensemble de sommets avec $v_0=v_{n+1}$ représentants le dépôt et $V'=V/\{v_0, v_{n+1}\}$ l'ensemble des clients.

$U = \{(v_i,v_j) / v_i,v_j \in V', i \neq j\}$ est l'ensemble des arêtes formant les tournées. Chaque tournée débute à $v_0$ et se termine à $v_{n+1}$.

Une matrice de coûts ou distances $c_{ij}$ entre les clients $v_i$ et $v_j$ (On pose $c_{0,n+1} = 0$).
*d* représente un vecteur des demandes clients.

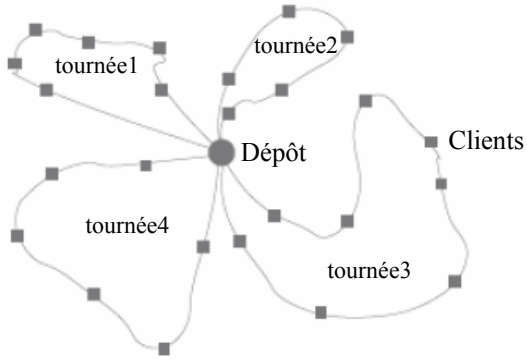


**Fig. 3.** Le problème des tournées de véhicules

$R_i$ est la tournée du véhicule *i* représentée par une permutation de clients visités par le véhicule. La flotte de véhicules $\mathfrak{I}$ est composée de véhicules tous identiques de capacité C et de coût fixe F. Un véhicule est assigné à chaque tournée. Ce modèle contient des variables de décision notées $x_{ij}^{\ell}$ (défini $\forall$ i,j $\in$ V, $\forall \ell \in \mathfrak{I}$) et prennent la valeur 1 si le véhicule $\ell$ va du client *i* au client *j*, 0 sinon. Le VRP peut donc être formulé comme suit :

$$\text{Min } Z = F\sum_{j\in V'}\sum_{\ell\in\mathfrak{I}} x_{0j}^{\ell} + \sum_{i\in V}\sum_{j\in V} c_{ij}\sum_{\ell\in\mathfrak{I}} x_{ij}^{\ell} \quad . \tag{1}$$

$$\sum_{i\in V}\sum_{\ell\in\mathfrak{I}} x_{ij}^{\ell} = 1 \qquad \forall\, j \in V' \,. \tag{2}$$

$$\sum_{j\in V}\sum_{\ell\in\mathfrak{I}} x_{ij}^{\ell} = 1 \qquad \forall\, i \in V' \,. \tag{3}$$

$$\sum_{j\in V'} x_{0j}^{\ell} = 1 \qquad \forall \ell \in \mathfrak{I} \quad . \tag{4}$$

$$\sum_{i\in V} x_{ik}^{\ell} - \sum_{j\in V} x_{kj}^{\ell} = 0 \qquad \forall\, k \in V', \forall \ell \in \mathfrak{I} \,. \tag{5}$$

$$\sum_{i\in V'} x_{i,n+1}^{\ell} = 1 \qquad \forall \ell \in \mathfrak{I} \,. \tag{6}$$

$$\sum_{i\in V'} d_i \sum_{j\in V} x_{ij}^{\ell} \le C \qquad \forall \ell \in \mathfrak{I} \,. \tag{7}$$

$$x_{ij}^{\ell} \in \{0,1\} \qquad \forall\, i, j \in V, \forall \ell \in \mathfrak{I} \,. \tag{8}$$

La fonction objectif (1) indique le coût total à minimiser et qui comprend le coût des véhicules et le coût des tournées. Les contraintes (2) et (3) indiquent que chaque client doit être affecté à exactement un véhicule. Les contraintes (4), (5) et (6) modélisent les contraintes de flux qui garantissent que chaque véhicule prend son départ du dépôt, puis visite au moins un client pour enfin retourner au dépôt. La contrainte (7) indique que la capacité des véhicules ne doit pas être dépassée.

Une solution réalisable est composée d'une partition $R_1,\ldots R_m$ de V, une permutation $\sigma_i$ de $R_i \cup 0$ indiquant l'ordre des clients servis par la tournée i. Une tournée $R_i$ est réalisable si le véhicule passe une seule fois par chaque client de la tournée sans dépasser la capacité de celui-ci.

Notre algorithme commence d'abord par la génération d'une solution réalisable initiale. Nous utilisons pour cela l'algorithme de Clark et Wright [7]. Puis, un ensemble de solutions est généré en utilisant les opérateurs de voisinage classiques tels que k-opt, or-opt, λ-échange [8, 17]. Pendant la génération de la population, un vocabulaire est constitué contenant l'ensemble des tournées réalisables trouvées dans la population. Pour réduire la taille du vocabulaire, nous utilisons un facteur de similarité des tournées. Par exemple, deux tournées sont considérées comme similaires si un certain nombre de clients sont visités par les deux tournées.

Comme dans la classification des documents, où un document est représenté par le nombre d'occurrences des mots dans un fichier benchmark, nous faisons de même pour une population de solutions. Un fichier benchmark est ainsi généré de manière similaire que les benchmarks LIBSVM [6] utilisés dans l'apprentissage machine. Une solution du VRP est représentée par une série de « $Id_{TR}$ : $Nb_{TR}$ » où $Id_{TR}$ représente le numéro de la tournée dans le vocabulaire et $Nb_{TR}$ le nombre d'occurrence de la tournée dans la solution. Après génération du fichier benchmark nous réalisons une classification non supervisée basée sur les USVM. Le résultat est un ensemble de clusters qui représentent des groupes de solutions de caractéristiques différentes. Enfin, nous générons une nouvelle population composée de solutions élites obtenues à partir des clusters.

Pour tester la capacité de notre implémentation à manipuler des problèmes de grande taille, nous avons utilisé un sous ensemble du corpus public SOLOMON comme décrit ci-dessous dans la table 1.

**Table 1.** Corpus SOLOMON

Classe : (R101~R104)

| Classe de véhicule | Capacité | Coût |
|---|---|---|
| A | 30 | 50 |
| B | 50 | 80 |
| C | 80 | 140 |
| D | 120 | 250 |
| E | 200 | 500 |

La plateforme utilisée dans notre expérimentation est un micro-ordinateur équipé d'un processeur Pentium D 1.6GHZ avec 512 Mo de mémoire vive fonctionnant sous Windows XP.

Les résultats de notre expérimentation comparé à la recherche tabou sont résumés dans la table 2 et montrent que notre algorithme peut effectivement traiter des instances de grande taille et trouver des solutions de haute qualité ce qui confirme la possibilité de développer des algorithmes « intelligents » qui peuvent automatiquement apprendre durant le processus de recherche.

**Table 2.** Résultats préliminaires

| Problème | Recherche tabou | Clustering |
|---|---|---|
| 1 | 5061 | 5061 |
| 2 | 5013 | **4998** |
| 3 | 4772 | 4774 |
| 4 | 4455 | **4358** |

## 5 Conclusion

Construire des algorithmes et des méthodes pouvant apprendre à partir de l'histoire de recherche a été pour longtemps un des plus grands challenges de l'intelligence artificielle. Nous avons introduit dans cet article un nouveau paradigme qui a pour objectif d'incorporer des techniques d'apprentissage machine tels que les SVMs dans des métaheuristiques à population de solutions. Nous présentons un algorithme intelligent qui peut apprendre à partir de l'histoire de recherche. Malgré un temps d'exécution assez conséquent, nous sommes convaincus que les avancées remarquables réalisés dans le domaine de l'apprentissage machine peuvent être profitable pour la résolution de problèmes d'optimisation combinatoire difficiles. Des expérimentations supplémentaires nous semblent nécessaires pour développer encore plus cette idée, objet de nos prochains travaux.

## Références


1. R. Battiti, M. Brunato, "Reactive Search for Traffic Grooming in WDM Networks", Sergio Palazzo (Ed.), Evolutionary Trends of the Internet, IWDC2001, LNCS2170, Springer, 2001.
2. R. Battiti, M. Brunato, "Reactive search: Machine learning for memory-based heuristics", Technical Report # DIT-05-058,Trento University (Italy), September 2005.
3. R. Battiti, F. Mascia, "Reactive and dynamic local search for max-clique: Does the complexity pay off?", Technical Report #DIT-06-027, 2006.
4. R. Battiti, M. Prostasi, "Reactive local for the maximum clique problem", In Springer-Verlag, editor, Algorithmica,volume 29, pages 610-637, 2001.
5. E. Breimer, M. Goldberg, D. Hollinger, D. Lim, "Discovering optimization algorithms through automated learning", In Graphs and Discovery, volume 69 of DIMACS Series in Discrete Mathematics and Theoretical ComputerScience, pages 7-27, American Mathematical Society, 2005.
6. C.C. Chang and C.J. Lin, "LIBSVM: a library for support vector machines", www.csie.ntu.edu.tw/~cjlin/libsvm.2003

7. G. Clarke, J. Wright, “Scheduling of vehicles from a central depot to a number of delivery points”. Operations Research 12(4), 568-581, 1964.
8. J.F. Cordeau, M. Gendreau, G. Laporte, J.Y. Potvin, F. Semet, “A guide to vehicle routing heuristics”. Journal of the Operational Research Society 53(5), 512-522, 2006.
9. Z. Ghahramani, “Unsupervised Learning”, GatsbyComputational Neuroscience Unit, University College London, UK, 2004.
10. F. Glover, “Heuristics for integer programming using surrogate constraints”, Decision Sciences, vol. 8, p. 156-166, 1977.
11. F. Glover, M. Laguna, “Tabu Search”. Kluwer AcademicPublishers, Amsterdam, Netherlands, 1997
12. F. Glover, M. Laguna, M. Rafael, “Scatter Search and Path Re-linking: Advances and Applications, In: F.Glover and Kochenberger (eds.): Handbook of Metaheuristics. (International Series in Operations Research & Management Science, vol. 57). Kluwer Academic Publishers, Boston, USA, 1–35, 2003.
13. J.H. Holland “Adaptation in Natural and Artificial Systems”, University of Michigan Press, Ann Arbor, 1975.
14. F. Hutter, Y. Hammadi, “Parameter Adjustment Based on Performance Predictions: Towards an Instance-Aware Problem Solver”, Technical Report, MSR-TR-2005-125,Microsoft Research, Cambridge, UK, 2005.
15. H. Mühlenbein, “Genetic algorithms”. In Aarts E., Lenstra J.K., Local Search in Combinatorial Optimization, John Wiley and Sons, p. 137-171, 1997.
16. J.P. Pedroso. “Tabu Search for Mixed Integer Programming”. Technical Reports Series DCC.2004
17. J.Y. Potvin, S. Bengio, “The vehicle routing problem with time windows”. INFORMS Journal on computing 8(2), 165-172, 1996.
18. I. Rechenberg, “Evolutionsstragtegie: Optimierung Technischer System nach Prinzipien der Biologischen Information”. Fromann, Freiburg, 1973.
19. E. Rodriguez-Tello, J.K. Hao, “Recherche tabou reactive pour le problème de l’arrangement linéaire minimum”, ORSA Journal on Computing, 2005.
20. O. Sammoud, S. Sorlin, C. Solnon, K. Ghédira, “A comparative study of Ant Colony optimization and Reactive Search for Graph Matching Problems”, 6th European Conference on Evolutionary Computation in Combinatorial Optimization, EvoCOP, 2006.
21. D. Schuurmans, “Discriminative, Unsupervised, Convex Learning”, MITACS Workshop, 2005.
22. S. Sorlin, C. Solnon, “Reactive tabu search for measuring graph similarity”, 5th IAPR Workshop on Graphbased Representations in Pattern Recognition (Gbr 2005), LNCS 3434 – Springer: 172-18, 2005.
23. V.N. Vapnik. “Statistical Learning Theory”. Wiley Inter-Science, 1998.
24. B. C. Williams, P. Kim, M. Hofbaur, J. How, J. Kennell, J.Loy, R. Ragno, J. Stedl, A. Walcott, “Model-based Reactive Programming of Cooperative Vehicles for Mars Exploitation”, In AI Magazine, 2001.