Test de cohérence des données pour un système expert d'aide au diagnostic de pannes

KARAOUZENE Zoheir
Département d'informatique. Ecole préparatoire en sciences et techniques.
Laboratoire IS2M. Université aboubekr belkaid
Tlemcen, ALGERIE
zoheir_karaouzene@yahoo.fr

CHEIKH Abdelmajid
Département de genie mécanique. Université Aboubekr Belkaid.
Laboratoire IS2M. Université aboubekr belkaid
Tlemcen, ALGERIE
am_cheikh@yahoo.fr

***Résumé* —Les systèmes d'aide au diagnostic de pannes matériels qui sont généralement des systèmes experts, permettent d'accélérer l'identification d'une panne donc de l'éviter ou de minimiser les dégâts matériels et humains. La plupart des recherches se sont focalisées uniquement sur la partie décision, mais la cohérence de la base de connaissance est un sujet important pour faire un raisonnement logique et valide, car les problèmes de l'incohérence peuvent donner des résultats incorrects ou des contradictions internes. Afin de remédier à de telles situations, nous proposons une méthode pour la vérification de la cohérence des données. Cette méthode utilise une approche de l'intelligence artificielle pour transformer le problème de test de cohérence en un problème de satisfiabilité (SAT). La mise en œuvre de la méthode proposée est réalisée à l'aide d'une adaptation de l'algorithme DPLL qui a été testée sur un cas réel dont les données sont fournies par une minoterie. Les résultats semblaient logiques et très encourageants.**

Mots clés— Système expert ; Base de connaissances ; Algorithme DPLL ; Problèmes SAT ; Diagnostic de pannes.

I. INTRODUCTION

Actuellement, l'enjeu principal dans le monde des entreprises de nos jours est la capacité à répondre efficacement en termes de temps et de qualité aux demandes des clients qui, au fils du temps, deviennent de plus en plus exigeants. La nécessité d'un bon système de production entretenu par un bon service de maintenance s'impose inévitablement. C'est ici qu'intervient les outils informatiques qui grâce à leur évolution rapide et constante au cours des dernières années joue un rôle essentiel et incontournable dans toutes les couches du monde des entreprises. Avec les progrès technologiques et informatiques les outils industriels sont de plus en plus complexes, l'environnement industriel change lui aussi d'où l'avènement de nouveaux concepts de maintenance. L'un de ces concepts est l'e-maintenance.

Notre problématique se trouve dans ce contexte et notre objectif est de proposer un outil de vérification de la validité des connaissances utilisées pour un système d'aide au diagnostic de pannes afin d'assurer un raisonnement logique. Le reste de cet article est organisé comme suit, la méthode proposée et son implémentation sont présentées dans les sections 2 et 3. La section 4 est réservée pour la discutions des résultats obtenues. Les conclusions et les futurs travaux de recherche sont résumés dans la dernière section.

Xème Conférence Internationale : Conception et Production Intégrées, CPI 2015, 2-4 Décembre 2015, Tanger - Maroc.

Xth International Conference on Integrated Design and Production, CPI 2015, December 2-4, 2015, Tangier - Morocco.

II. PRESENTATION DE LA METHODE PROPOSEE

Certains systèmes experts ne nécessitent pas d'actualiser ou de faire des mis à jour de leur bases de connaissances comme les jeux, les logiciels de la réalité virtuelle ou de la simulation virtuelle. Mais quand on parle des connaissances incomplètes dans un environnement qui change comme dans notre cas, les experts de maintenance doivent ajouter au fur et à mesure de nouvelles données à condition d'assurer la cohérence de ces données. Manuellement ce test est difficile presque impossible à cause de la taille et du type complexe des informations enregistrées. Nous allons montrer dans cette section comment automatisé le test de cohérence des données de la base de connaissances, c'est-à-dire comment transformer le problème de test de cohérence en un problème de satisfiabilité (SAT).

A. Un apperçu sur les Méthodes classiques de test de cohérence

Généralement, les algorithmes de diagnostic utilisent les données rangées dans une structure appelée : table de décision. Plusieurs algorithmes sont décrits dans [1,2,3,4,5].

Un exemple d'une table de décision est indiqué dans le tableau 1 pour un cas d'une minoterie. Les rangées de la table de décision représentent les cas, tandis que les colonnes représentent les symptômes {A,B,C,D }. Les pannes {p1,p2,p3,p4}, pour les machines doseur, mouilleur, brosse et séparateur, cette partie est appelée la partie décision.

Un ensemble de données est cohérent est celui qui ne contient pas des cas de conflit. Les cas de conflit sont les cas pour lesquels toutes les valeurs des attributs (dans notre exemple, les attributs sont symptômes) sont les mêmes, les valeurs de décision sont différentes. Les cas 1 et 8 représentent un exemple de conflit, on remarque que pour les mêmes symptômes, il existe des valeurs de décision différentes, donc cette table est incohérente.

Ce type d'algorithmes est efficace pour faire un diagnostic médical. Mais dans le domaine industriel il est moins adapté car les données sont plus complexes et elles appartiennent à un nombre important de machines, les symptômes ne sont pas les mêmes entre une machine est une autre, ce n'est pas le cas

dans le domaine médical ou les symptômes pour une maladie sont les mêmes pour toutes les personnes. Nous proposons dans cet article une méthode pour remplacer cette représentation par une représentation logique simple à manipuler et de transformer le problème de test de cohérence en un problème de satisfiabilité.

TABLEAU I : TABLE DE DECISION.

	machines	symptômes				pannes			
Cas		A	B	C	D	P 1	P 2	P 3	P4
1	Doseur		X	X				X	
2	Séparateur		X			X			
4	Doseur			X					X
5	Brosse	X							
6	Séparateur		X						
7	Mouilleur				X		X		
8	Doseur		X	X		X	X		X
A : Des graines de blé dans les déchets									
B : Corps étranglé dans la machine									
C : La vise tourne vers l'extérieure									
D : Le moteur est bloqué									
P1 : Vibrateur défectueux									
P2 : Le mixeur est bloqué									
P3 : Le joint est défectueux									
P4 : La vise du doseur est défectueuse									

B. Les problèmes de satisfiabilité

Il est possible de transformer certains problèmes de l'intelligence artificielle et utiliser les algorithmes de satisfiabilité (SAT) pour résoudre efficacement ces problèmes. Gu Purdom et al [6] ont écrit un article citant de nombreuses applications de problèmes de satisfiabilité dont certains peuvent être classés comme suit:

- En informatique et l'intelligence artificielle comme: le problème de satisfaction de contraintes [7,8], la programmation logique [9,10], la sémantique d'information [11,12], la maintenance et diagnostic de panne [13,14] , et les systèmes de production [15,16].
- En fabrication assistée par ordinateur comme: la planification de tâches [17], la conception [18,19] et la modélisation et configuration robuste de tâches [20].

C. La représentation de données de la base de connaissnaces

La représentation des connaissances est un sujet important dans le domaine de l'intelligence artificiel. Juan Acosta Guadarrama et al [21] ont proposé de transformer les données en clauses logiques pour la mise en œuvre d'un débogueur de base de connaissances. Cette représentation est très adaptée pour résoudre les problèmes de satisfiabilité puisqu'elle est sous une forme logique. Dans ce travail, La logique d'ordre 0 a été choisie pour la représentation des connaissances, donc les données seront encodées sous forme de propositions.

En logique mathématique:

- Une proposition est une expression logique qui peut être vraie ou fausse.

  Exemple 1 : la proposition $a : s \vee t \rightarrow q$ désigne que si s ou t alors q.
- Une forme normale conjonctive (FNC) est une normalisation d'une expression logique qui est une conjonction de clauses

  Exemple 2 : la proposition a de l'exemple 1 s'écrit en (FNC) comme suit : $a : (\neg s \vee q) \wedge (\neg t \vee q)$
- Une clause est une disjonction de littéraux.
- Un littéral est un atome (également appelé littéral positif) ou la négation d'un atome (également appelé littéral négatif).
- Un atome est une variable propositionnelle.

Pour rendre le type de connaissance compatible avec les problèmes de satisfiabilité, nous proposons de transformer les propositions en clauses logiques.

D. Le Principe de resolution:

Le principe de résolution est une méthode automatique pour montrer la validité d'une formule. Intuitivement, dire que A est une conséquence logique de B c'est dire que A est nécessairement vraie lorsque B est vraie.

Pour appliquer ce principe, il faut d'abord transformer la formule en sa forme normale conjonctive, ensuite éliminer les connecteurs. On obtient ainsi un ensemble S de clauses.

Exemple :

Considérant les clauses:

$C_1 : L_1 \vee C_1^{'}$ et $C_2 : \neg L_2 \vee C_2^{'}$.

Avec: L_1 *et* L_2 sont deux littéraux et $C_1^{'}$ et $C_2^{'}$ sont deux clauses qui peuvent être vide. Si $L_1 = L_2$ La clause $C : C_1^{'} \vee C_2^{'}$ est la disjonction des clauses restantes après suppression des littéraux L_1 *et* L_2 de C_1 et C_2 . La clause C est appelée la clause résolvante. L_1 *et* L_2 sont les littéraux résolus.

Formellement :

$$C_1 , C_2 \underset{\text{Résolution}}{\vdash} C_1^{'} \vee C_2^{'}$$

Par définition, $C : C_1^{'} \vee C_2^{'}$ est vraie si C_1 et C_2 sont varient et on dit que $C : C_1^{'} \vee C_2^{'}$ est une conséquence logique de C_1 et C_2

La clause vide est notée par le symbole: /////.

Exemples :

- La clause résolvante de $C_1 : \neg p \vee q$ et $C_2 : p \vee r$ est $q \vee r$.

  formellement : $C_1, C_2 \underset{\text{Resolution}}{\vdash} q \vee r$

- On remarque que: $p \vee r$ est la clause résolvante de C_1 et C_2.

- $C_3 : \neg p \vee q$ et $C_4 : p \vee \neg q$ sont deux clauses contenants des littéraux complémentaires. donc la clause résolvante est la clause vide. Formellement: $C_3, C_4 \underset{\text{Resolution}}{\vdash} //////$.

Dans ce cas, l'ensemble de clauses S formé de C_3 et C_4 est un ensemble contradictoire ou insatisfiable. Par le principe de résolution, un ensemble de Clauses S est insatisfiable si S mène par résolution à la clause vide et on écrit:

$$S \underset{\text{Resolution}}{\vdash} /////$$

Ce principe va être utilisé pour le test de cohérence.

E. *Le test de coherence*

Rappelant que la base de connaissance est codée sous la forme clausale. Si on suppose que l'ensemble de clauses S représente la base de connaissances, S est insatisfiable cela veut dire que S est incohérent. Donc deux cas sont possibles :

$$\begin{cases} Si \quad S \underset{\text{Resolution}}{\vdash} ///// \quad S \text{ est incohérent} \\ Sinon \quad S \quad \text{est Cohérent.} \end{cases}$$

L'idée est de vérifier par l'application du principe de résolution si un ensemble C composé d'une ou plusieurs clauses correspond à une nouvelle information à ajouté à la base de connaissances S initialement satisfiable assure toujours sa validité. Les deux cas possibles sont :

$$\begin{cases} Si \quad S \cup C \underset{\text{Resolution}}{\vdash} ///// \quad \text{alors C rejeté} \\ Sinon \quad \text{l'ajout de l'ensemble C à S est possible} \end{cases}$$

Donc l'ajout de C à S n'est possible seulement si C ne cause pas une contradiction avec l'ensemble de clauses de S. la figure 1 illustre le schéma général de la technique présentée

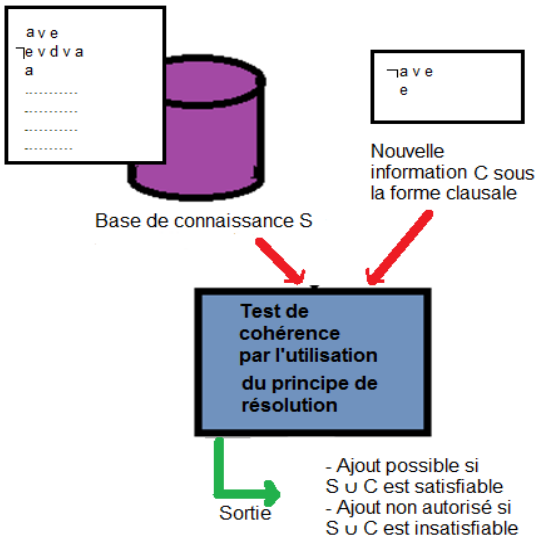

Fig.1 Le principe du test de cohérence.

Nous avons montré comment transformé le problème de cohérence de connaissances en un problème de satisfiabilité. L'implémentation du solveur SAT est présentée dans la section suivante.

III. IMPLEMENTATION DE LA METHODE

L'algorithme de Davis Putnam Logemann Loveland (DPLL) présenté dans [22], [23] est efficace pour résoudre les problèmes de satisfiabilités. C'est un algorithme de backtracking complet, de résolution du problème SAT exprimé en forme normale conjonctive. L'algorithme permet de vérifier est ce qu'une clause C est cohérente avec toutes les autres formules du système de clause S, ce test est appliqué de manière récursive sur toutes les clauses de la base de connaissances. Cet algorithme à été adapté pour implémenter la partie inférence de la méthode proposée.

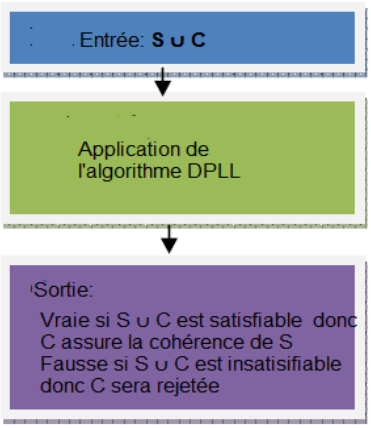

Fig.2 Adaptation de l'algorithme DPLL.

Le nombre de données sous forme de clauses enregistré dans la base de connaissances qui peut être important nous impose d'utiliser la structure chainée. La base de connaissances est une liste chainée des éléments de type clause. Une clause est une autre liste chainée dont les éléments

sont des structures à deux cases pour représenter un littéral, la première case est réservée pour la variable propositionnelle est la deuxième pour le signe de la variable. Pour faciliter la manipulation des clauses, nous avons choisi de remplacer la négation par le signe (-) et le signe (+) si la variable ne contient pas de négation. La figure 3 est un exemple d'une base de connaissance nommée S composé de trois clauses A, B et C avec :

$A : a \vee \neg e$

$B : \neg d$

$C : e$

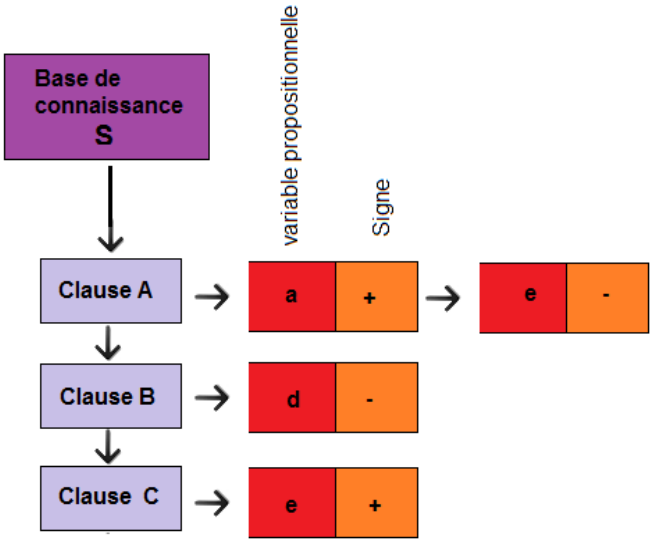

Fig.3 La représentation de la base de connaissance en mémoire de travail

Donc la nouvelle écriture du S est :

A : +a –e.

B : -d.

C : +e

On remarque que le système de clause S est cohérent. La nouvelle clause D : -a+e rendre le système S incohérent puisque A et D sont deux clauses contradictoires. Le principe général de l'algorithme DPLL est de chercher de manière récursive ce type d'incohérence. Si deux clauses ont les mêmes variables propositionnelles mais avec des signes opposites donc l'algorithme juge le système s incohérent.

IV. TEST DE LA METHODE ET LES RESULTATS OBTENUS

Le tableau 1 illustre les résultats de l'application de la méthode proposée sur une base de connaissances initiale sous forme de 14 propositions spécifiques aux quatre machines de la minoterie de Mahdia-Tiaret ALGERIE. Ces données ont été utilisées par Zoheir KARAOUZENE et al [24] comme une base de connaissances d'un système expert d'aide au diagnostic de pannes. Pour l'insertion de nouvelles formules à la base initiale, le logiciel développé a autorisé l'ajout de 7 formules et il a refusé 3 autre formules pour raison de leurs incohérences.

TABLEAU II. LES RESULTATS.

Machine	Formulas
Séparateur	*Littéraux:* Suppresseur marche : Sp_On ; Vibrateur marche : Vib_On ; Corps étranglé dans le séparateur : Body_St ; Courant passe: Curr_Pass.
	La base de connaissances initiale: $1 : \neg Sp\_On \rightarrow (\neg Vib\_On \vee Body\_St)$ $2 : Sp\_On \rightarrow Vib\_On$ $3 : Vib\_On \rightarrow Curr\_Pass$
	Formules ajoutées: $15 : \neg Curr\_pass \rightarrow \neg Sp\_On$ $16 : Sp\_On \rightarrow Curr\_Pass$
	Formule rejetée $A : \neg Body\_st$
Doseur	*Littéraux:* La vise du doseur est bloquée: Scr_Bk ; le joint du doseur est défectueux : Seals_Def la vise tourne vers l'extérieure: Scr_Out la vise tourne vers l'extérieure : $\neg Scr\_Out$ La base de connaissances initiale: $4 : Scr\_Bk \rightarrow Seals\_Def \vee Body\_St$ $5 : (Seals\_Def \wedge Scr\_Out) \rightarrow Body\_St$ $6 : (Scr\_Bk \wedge \neg Scr\_Out) \rightarrow Seals\_Def$
	Formules ajoutées $17 : (Scr\_Bk \wedge \neg Body\_St) \rightarrow Scr\_Out$ $18 : (Scr\_Bk \wedge \neg Body\_St) \rightarrow Seal\_Def$

Brosse	*Littéraux:* Des graines de blé dans les déchets : Sd_w ; le tamis est bloqué: Sieve_Bk ; La brosse est bloquée: Br_Bk ; le moteur est bloqué : Mt_Bk ; Courroie est défectueuse : Belt_df ; La base de connaissances initiale: $7: Sd\_w \rightarrow Sieve\_Bk$ $8: Br\_Bk \rightarrow (Mt\_Bk \vee Beld\_Df \vee Body\_St)$ $9: (Br\_Bk \wedge \neg Mt\_Bk \wedge \neg Belt\_Df) \rightarrow Body\_$ $10: (Br\_Bk \wedge \neg Mt\_Bk) \rightarrow Belt\_Df$ $11: \neg Belt\_Df \rightarrow \neg Sieve\_Bk$
	Formule ajoutée $19: \neg Br\_Bk \rightarrow \neg Mt\_Bk$
	Formule rejetée: $B: \neg Belt\_Df \wedge Sieve\_Bk$
Le mouilleur	*Littéraux:* Le mouilleur est bloqué: Moist_Bk ; Le mixeur est bloqué: Mix_Bk ; La base de connaissances initiale: $12: Moist\_Bk \rightarrow (Mt\_Bk \vee Sieve\_Bk \vee Mix\_B$ $13: \neg Moist\_Bk \rightarrow \neg Mt\_Bk$ $14: \neg Mix\_Bk \rightarrow \neg Sieve\_Bk$
	Formules ajoutées $20: \neg Moist\_Bk \rightarrow \neg Belt\_Fd$ $21: Mt\_Bk \rightarrow Mix\_Bk$
	Formule rejetée: $C: Mt\_Bk \wedge Beld\_Fd$

La première formule rejetée est $A: \neg Body\_st$ désigne que l'existence d'un cops étranglé dans le séparateur est impossible quelque soit l'état du séparateur ceci cause une incohérence avec les formules 1 et 2, ces deux formules indiquent qu'il y'a une possibilité que le séparateur sera bloqué à cause de l'existence des corps étranglés. De la même manière, la deuxième formule B cause une incohérence avec la formule 11 et la formule C cause une autre incohérence avec les formules 13 et 20.

La Figure 4 est une prise d'écran d'une partie du logiciel développé qui permet à l'utilisateur d'insérer des connaissances sous forme de propositions pour chaque machine et d'effectuer le test de cohérence de la nouvelle base de connaissances.

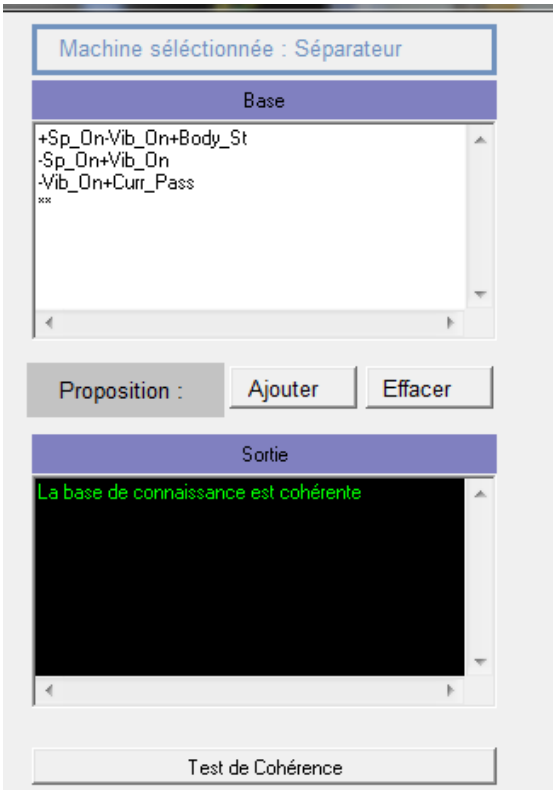

Fig.4 Une partie de l'interface de l'application.

V. CONCLUSION

L'ajout de nouvelles données à la base de connaissances est une fonction délicate, elle peut être dangereuse si elle ne sera pas précédée par un test de sa cohérence qui est manuel dans plusieurs systèmes. Dans cet article une méthode est proposée pour automatiser ce test qui est basé sur les principes de la logiques des propositions consistant a transformer le problème de test de cohérence de données en un problème de satisfiabilité. Ce test est facilement résolu par une simple adaptation de l'algorithme DPLL. Les futurs travaux seront focalisés sur la réduction de la complexité de l'algorithme DPLL qui est NP-Complet, par la proposition des heuristiques qui seront combinés avec l'algorithme DPLL.

REFERENCES

[1] Z. Pawlak, Rough Sets, Theoretical Aspects of Reasoning about Data, Kluwer Academic Publishers, Dordrecht, Boston, London, 1991.

[2] Z. Pawlak, A. Skowron, Rough sets: some extensions, Inf. Sci. 177 (2007) 28–40.

[3] Z. Pawlak, A. Skowron, Rudiments of rough sets, Inf. Sci. 177 (2007) 3–27.

[4] P. Samanta, M.K. Chakraborty, On extension of dependency and consistency degrees of two knowledges represented by covering, Trans. Rough Sets 9(2008) 351–364.

[5] Z. Suraj, W. Rzasa, Decision system analysis according to the relaxed consistency principle, Int. J. Inf. Technol. Intell. Comput. 1 (2006) 113–125.

[6] GU J., PURDOM P., FRANCO J., WAH B., « Algorithms for the Satisfiability problem : a survey », DIMACS Series on Discrete Mathematics and Theoretical Computer Science, American Mathematical Society 35, p. 19–151, 1997.

[7] J. A. Anderson and G. E. Hinton. Models of information processing in the brain. In G. E. Hinton and J. A. Anderson editors, Parallel Models of Associative Memory, chapter 1, pages 9- 48 Lawrence Erlbaum Associates, Publishers, Hillsdale, New Jersey, 1981.

[8] J. Gaschnig. Performance Measurements and Analysis of Certain search Algorithms. PhD thesis, Carnegie-Mellon University, Dept of Computer Science, May 1979.

[9] J. Cohen, editor. Special Section on Logic Programming. Comm. Of the ACM, volume 35, number 3. 1992.

[10] A, Colmerauer. Opening the Prolog III universe. BYTE Magazine, page 177-182, Aug. 1987.

[11] T.A. Marsland and M. Campbell. Parallel search of strongly ordered game trees. ACM Computing Surveys, 14(4):533-551, Dec. 1982.

[12] T.A. Marsland and J. Schaeffer. Computers, Chess, and cognition. Springer-Verlag, New York, 1990.

[13] J. de Kleer. Exploiting locality in a TMS. In Proceedings of A A A I'90, pages 264-271, 1990.

[14] R. Dechter. A constraint- network approach to truth maintenance. Technical Report R-870009, Computer Science Dept., UCLA, Los Angeles, 1987.

[15] T. Ishida. Parallel Rule firing in production systems. IEEE Trans. on Knowledge and Data Engineering, 3(1):11-17, Mar.1991.

[16] F. Jahanian and A.K. Mok. Safety analysis of timing properties in real-time systems. IEEE Trans. on software Engineering, SE-12(9):890-904, Sept. 1986.

[17] B.A. Nadel and J. Lin. Automobile transmission design as a constraint satisfaction problem: A collaborative research project with ford motor Co. Technical report, Wayne Sate Iniversity, 1990.

[18] D. Navinchandra, Exploration And Innovation in Desingn. Springer Velag. Sadeh, 1990.

[19] D . Navinchandra and D.H. Marks. Layout planning as a consistent labeling optimization problem. In Proceeding of 4th international symposium on Robotics and AI in constructin, 1987.

[20] F. Frayman and S. Mittal. Cossack: A Constraints- based Expert system for Configuration Tasks. Computational Mechanics Publications, Nadel, 1987.

[21] Juan C. Acosta Guadarrama, J. Raymundo Marcial Romero, Marcelo Romero,Jorge Hern´andez Camacho , Implementing a Knowledge Bases Debugger , Journal of computer society IEEE DOI 10.1109/MICAI. 2012.

[22] Renato Bruni and Andrea Santori, New updating criteria for conflict-based branching heuristics in DPLL algorithms for satisfiability. Science direct journal. on Discrete Optimization 5 (2008) 569–583.

[23] Peter Baumgartner and Cesare Tinelli, The model evolution calculus as a first-order DPLL method. ScienceDirect journal. on Artificial Intelligence 172 (2008) 591–632.

[24] Z. Karaouzene et A. Cheikh. A method for transforming the problem of fault diagnosis in a problem of satifiability (SAT): Case of an industrial mill. Dans le proceeding International Conference on Advanced Technology & Sciences (ICAT'15) , Antalya – TURQUIE 2015.