REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université de M'sila
Faculté des Mathématique et de l'Informatique
Département de l'Informatique

Mémoire de fin d'études Pour l'obtention Diplôme de
Master Académique en Informatique

Domaine : Mathématique informatique

Filière : Informatique fondamentale

Option : Système d'information avancé

Thème :

L'application de colonie de fourmis au problème d'ordonnancement des instructions

Réalisé par :

BEN GACEMI Afaf

Diriger par :

Mr. BOUAMAMA Salim

Mr. GASMI Abdelkader.

Promotion : 2010/2011

Table des Matières

INTRIDUCTION GENERALE 1

Chapitre 1 – L'optimisation des Problèmes combinatoires

1.1 Introduction 4
1.2 Généralités sur les problèmes d'optimisation 4
1.2.1 Problème SAT 4
1.2.2 Coloration de Graphe 4
1.2.3 Voyageur de commerce (TSP) 5
1.2.4 Problème d'affectation quadratique (QAP) 5
1.3 Généralisation des définitions 5
1.3.1 Problèmes d'optimisation 5
1.3.2 Classes de complexité 6
1.3.3 Fonction objectif 6
1.3 Méthodes de résolutions 7
1.3.1 Classifications des méthodes de résolution 7
1.3.2 La démarche heuristique 9
1.3.3 Les méta-heuristiques 9
1.3.3.1 Organisation générale 11
1.3.3.2 Applications 11
1.3.3.2.1 Méta-heuristique à recuit simulé 11
1.3.3.2.1 Les méta-heuristique évolutionnaire/ génétiques 12
1.3.3.2.1 LES méta-heuristiques éthologiques/colonies de fourmis 13
1.4 Conclusion 15

Chapitre 2 - Problème Traité: L'ordonnancement des instructions

2.1 Introduction 17
2.2 Exemple introductif 17
2.3 La notion de parallélisme au niveau d'instructions(ILP) 18
2.3.1 Définition 18
2.3.2 Techniques de base d'ILP en architecteurs des ordinateurs 19
2.3.2.1 Architecture pipeline 19

2.3.2.2 Architecture superscalaire 22
2.3.2.3 Les architectures VLIW 22
2.3.3 Des defis 23
2.3.4 Solusion 23
2.4 L'ordonnancement des instructions 24
2.4.1 Les types d'ordonnancement des instructions 24
2.5 Dépendances de données 24
2.5.1 Définition 24
2.5.2 Les types de dépendances 23
2.5.2.1 Dépendance de flot 24
2.5.2.2 Anti-dépendance 25
2.5.2.3 Dépendance de sortie 25
2.5.2.4 Dépendance d'entrée 25
2.5.3 Exemple 24
2.5.4 Représentations de dépendance de donnée 26
2.5.4.1 Exemple 26
2.6 L'ordonnancement des instructions et les méthodes heuristique 27
2.6.1 Formulation de problème d'ordonnancement des instructions local 27
2.6.1.1 Instance 27
2.6.1.2 L'ordonnancement faisable 27
2.6.1.3 But 27
2.6.2 L'heuristique et l'ordonnancement des instructions 28
2.6.2.1 La complexité du problème et les priorités heuristiques 28
2.6.2.2 L'algorithme de liste 29
2.6.2.3 Exemple 31
2.7 Conclusion 33

Chapitre 3 - Les colonies de fourmis, L'ordonnancement des instructions

3.1 Introduction 35
3.2 Comportement des fourmis réelles 35
3.2.1 L'expérience à doubles branches 35
2.2.2 L'expérience de l'obstacle 36
3.3 Les fourmis artificielles 37
3.4 L'optimisation par colonie de fourmis (ACO) 39

3.4.1 Représentation de problème 39
2.4.1.1 Définition formelle d'un problème d'optimisation 39
2.4.1.2 Définition formelle d'un problème d'optimisation 39
3.4.2 Les effets de l'algorithme ACO 41
3.4.2.1 ConstructAntSolutions 41
3.4.2.2 Les actions démon 41
3.4.2.3 L'évaporation de phéromone 41
3.5 Résolutions quelque problème d'optimisation combinatoire 41
3.5.1 Voyageurs de commerce(TSP) 41
3.5.2.1 Description générale 41
3.5.2.2 Résolution par l'Ant System (AS) 42
3.5.1 Optimisation des tables de routage 43
3.5.2.1 Description générale 43
3.5.2.2 Principe de l'algorithme 44
3.6 L'application des colonies de fourmis au problème d'ordonnancement des instructions . 45
3.6.1 Formulation d'algorithme (IS) 45
3.6.2 *Max-Min* Ant System pour l'ordonnancement des instructions 46
3.6.2.1 Présentation de MMAS 46
3.6.2.2 Contraintes sur les valeurs de phéromones 46
3.7 Conclusion 47

Chapitre 4 - Implémentation et des résultats expérimentaux

4.1 Introduction 49
4.2 Plate forme et outils de développement 49
4.3 Implémentation 50
4.3.1 Les caractéristique des benchmarks utilisés 50
4.3.2 Codage de donnée 51
4.3.2.1 Représentation d'un graphe 52
4.3.2.2 Codage des soluions 52
4.3.2.3 L'algorithme de liste (liste scheduling) 52
4.3.2.4 L'algorithme Max-Min Ant System 52
4.4 Expérimentation 53
4.4.1 La machine virtuelle ILOC 53
4.4.2 Tests et résultats 55
4.5 Conclusion 61

CONCLUSION GÉNÉRALE ET PERSPECTIVES 63
RÉFÉRENCES BIBLIOGRAPHIQUE 66
ANNEX 69

INTRODUCTION GÉNÉRALE

La vitesse d'exécution des programmes séquentiels sur les architectures modernes est sensible à l'ordre dans lequel les instructions sont présentées au processeur. Les dépendances entre les instructions posent un obstacle majeur pour l'exploitation du parallélisme de niveau instruction (ILP : Instruction-Level Parallelism) qui est un parallélisme de grain très fin par opposition au parallélisme de tâche ou au parallélisme entre itérations de boucles. C'est un moyen qui permet d'augmenter le degré du parallélisme en exécutant plus d'une instruction à la fois. ILP est une famille des processeurs et des techniques de conception du compilateur qui accélèrent l'exécution en causant des opérations (de style RISC) telles que la multiplication, la division et l'accès mémoire de s'exécuter en parallèle individuellement sur différentes unités fonctionnelles.

Depuis que l'architecture ILP exploite le parallélisme implicite que la majorité des programmes le contiennent, leur performance dépend fortement de l'ordre d'exécution des instructions (l'ordre lexical) dans la séquence du code. Si les instructions consécutives ont des dépendances de données, du contrôle ou des contraints ressources alors des latences surviennent et la performance diminue.

Par conséquent, l'exploitation efficace du parallélisme implicite entre les instructions exige le changement d'ordre des instructions par le compilateur (réordonnancer) sans affecter la sémantique du programme afin de réduire ou éviter les aléas de données (les dépendances de hasard). La tâche d'ordonnancement des instructions (Instruction Scheduling) occupe menant une place importante dans le processus d'optimisation de bas niveau pour les compilateurs modernes. Les techniques d'ordonnancement pour l'architecture ILP peuvent être subdivisées en deux types : local et global.

Le problème d'ordonnancement d'instructions est reconnu dans la littérature comme un problème difficile à résoudre dans le domaine de l'optimisation combinatoire; sa complexité est de type NP-Complet. Par conséquent, il n'existe pas un algorithme connu qui garanti une solution optimal dans un temps polynômial. Pour cette raison, la majorité des méthodes de résolution en pratique fait appel à des méthodes approchées (heuristiques) afin de construire des solutions de bonne qualité avec un temps raisonnable de calcul. La plupart des ordonnancements utilisés dans un bloc de base par les compilateurs sont basés sur les algorithmes de liste (List Scheduling) .C'est un algorithme glouton qui construit une liste d'instructions (un ordonnancement) en rajoutant à la fois et au fur et à mesure une instruction

sans prédécesseurs ou libérée par l'exécution d'autres instructions selon une priorité heuristique associée a chaque instruction. Afin d'améliorer le fonctionnement de l'algorithme de liste, une tentative de combiner plusieurs priorités heuristiques dans une seule priorité méta-heuristique simple a été étudiée en utilisant l'application de colonie de fourmis idée.

Les travaux précédents dans le domaine d'ordonnancement ont montré que l'application de colonie de fourmis a été capable de trouver de bonne qualité de solutions. Dans notre travail vous avez appliqué une variante de l'algorithme de colonies de fourmis nommée Max-Min Ant System (MMAS) pour ce problème, nous n'avons pas utilisé l'algorithme de liste comme une méthode de recherche local mais seulement le même parcours effectué par l'algorithme de liste sera suivie par les fourmis artificielles en expirant qu'ils vont produire des solutions meilleures ou équivalentes que le travail précédant en termes de qualité des solutions et temps d'exécution.

Nous organisons notre propos en quatre chapitres principaux :

- Le premier chapitre, est un état de l'art des problèmes d'optimisations combinatoire et des techniques de résolutions, axé sur les méta- heuristique.
- Ensuite, dans le second chapitre, nous décrivons des définitions liées a le problème d'ordonnancement des instructions) et les méthodes heuristiques pour résolus ce problème (list scheduling) avec un exemple.
- Le troisième chapitre est une présentation de l'optimisation par colonie de fourmis.
- Le dernier chapitre est constitué d'une implémentation de deux algorithmes utilisés (L'algorithme de liste et MMAS) et un petit teste.

CONCLUSION GÉNÉRALE ET PERSPECTIVE

Dans ce mémoire nous avons étudié le problème d'ordonnancement des instructions qui appartient à la classe des problèmes NP-complets. La résolution de ce problème par les méthodes exactes est plus difficile et surtout pour les blocs de base très grand (représentation de graphe). Le recours aux métaheuristiques est donc impératif.

Dans notre travail, nous avons implémenté les deux algorithmes ; l'algorithme de liste (list scheduling) et MAX-MIN Ant System pour ce problème.

- l'algorithme de liste : la plupart des ordonnancements utilisés dans un bloc de base par les compilateurs sont basés sur les algorithmes de liste (List Scheduling). C'est un algorithme glouton qui construit une liste d'instructions (un ordonnancement) en rajoutant à la fois et au fur et à mesure une instruction sans prédécesseurs ou libérée par l'exécution d'autres instructions selon une priorité heuristique associée à chaque instruction.

  On peut citer comme priorités : CP/MISF (Critical Path/Most Immediate Successors First), LP (Longest Path), HLF (Highest Level First), Highest Co-level First (HCF), LNSF (Largest Number of Successors First), LPTF (Largest Processing Time First), GMS (Gibbons and Muchnick scheduer), ainsi que d'autres heuristiques spécifiques basées sur l'algorithme de liste.
- Max-Min Ant System : cette variante (notée MMAS) est fondée sur l'algorithme AS et présente quelques différences notables:
  1. Seule la meilleure fourmi met à jour une piste de phéromone;
  2. Les valeurs des pistes sont bornées par t_{min} et t_{max};
  3. Les pistes sont initialisées à la valeur maximum t_{max}.
  4. La mise à jour des pistes se fait de façon proportionnelle, les pistes les plus fortes étant moins renforcées que les plus faibles;
  5. Une réinitialisation des pistes peut être effectuée.

Les meilleurs résultats sont obtenus en mettant à jour la meilleure solution avec une fréquence de plus en plus forte au cours de l'exécution de l'algorithme.

Après implémentation et analyse des deux algorithmes, on a pu constater que les résultats des deux approches sont très voisins, avec un léger succès de l'algorithme MMAS sur l'algorithme de liste scheduling.

Pour cela et afin de compléter notre travail, et comme perspectif, nous proposons de s'investir dans la recherche d'une nouvelle heuristique locale plus performante pour les algorithmes ACO.

Références Bibliographiques

Références bibliographiques

[1] B.Gasbaoui, "*optimisation de l'énergie réactive dans un réseau électrique»,* mémoire de magistère, univ Bachar, Algérie, Format HTML, disponible sur : http://www.memoireonline.com/04/11/4389/m_Optimisation-de-lenergie-reactive-dans-un-reseau-denergie-electrique34.html.

[2] B.Abdou, "*deux approches parallèles basées sur les colonies de fourmis pour la réalisation du problème de T-coloration des graphes*", mémoire de fin d'étude, ESI oued-smar, Alger, Format PDF, disponible sur : http://share.esi.dz/index.php?option=com_docman&task=cat_view&gid=65&Itemid=1

[3] R. Faure, "*Guide de la recherche opérationnelle*",2 tomes, Masson, 1990, pp 197-198.

[4] C. Papadimitriou, *"Computational Complexity*". Addison Wesley, 1994, pp 261,265.

[5] W. Stallings,"*Organisations et Architecture de l'ordinateur*",6 édition, Pearson, 2003, pp 256-260.

[6] D. Culler, J. Singh, A. Gupta. "*Architecture d'ordinateur parallèle - une approche de matériel/logiciel*". Morgan Kaufmann Publisher, 1999, pp 15.

[7] Y. Patt. "*Le microprocesseur Dix ans dès maintenant : Quels sont les défis, comment nous les rencontrent*", Entretien distingué de conférencier à Université de Carnegie-Mellon, Avril 2004. Recherché dessus 7 novembre, 2007.

[8] J. Hennessy and D. Patterson, "*Computer Architecture: A Quantitative Approach*", 4th edition, Morgan Kaufman Publishers, San Francisco, 2007, pp 66-78.

[9] N. Hong San, "*TIPE - La détection du parallélisme dans des programmes informatique*", Promotion 10, 15 juillet 2005.pp 6-7-8.

[10] S. Bouamama, "*Local Instruction Scheduling using Genetic Algorithms* ".Jordan University of Science and Technology, Irbid, Jordan.

[11] N.Ayari, "*Métaheuristiques parallèles hybrides pour l'optimisation combinatoire : problème de règles de Golomb*" pp 9- 13, Format PDF, disponible sur : http://ayarinaouel.webs.com/Master%20Thesis.pdf

[12] G. D. DUPLESSIS and P.MYCEK, "*Ant Colony Optimization* "INSA de Rouen, Département Génie Mathématique.

[13] J.Dréo and A.pétrowski, " *Métaheuristiques pour l'optimisation difficile* ", édition Eyroless, 2003, pp 120-121.

[14] J.L. Hennessy and D.A. Patterson," *Computer Architecture: A Quantitative Approach*", 3th edition, Morgan Kaufman Publishers, San Francisco, 2007, pp 221-222-223.

[15] E.GUERROUT, " *Optimisation des Protocoles de Routage avec la méthode de colonie de fourmis*", mémoire d'ingéniera d'état, Institut National de formation en Informatique (*I.N.I*) Oued-Smar, Alger, Promotion: 2006/2007, pp29-33.

[16] A. Costanzo, T. Luong and G. Marill," *Optimisation par colonies de fourmis*", 2006, pp 4-10.

[17] T. Risset, F.Dinechin,"*Cours de Compilation*", ENS-Lyon, Master 1 Informatique, 2005-2006.

[18] K. Cooper, P. Schielke, and D. Subramanian, "An *Experimental Evaluation of list scheduling*", Technical report TR98-326, Rice University, September 1998.

[20] D. Houzet, R. Chouvance, "*Microprocesseurs Architecture et performances*", article E 3 555, Techniques de l'Ingénieur, traité Électronique.

[21] A. Malik, T. Russell, M. Chase, and P. Beek, "*Learning Heuristics for Basic Block Instruction Scheduling*», School of Computer Science, University of Waterloo, Waterloo, Canada.

[22] L. Mugwaneza "*Architecture des ordinateurs"* article ESIL/Dépt. Informatique, A3-SICA- 2010/2011.

[23] www. BENCHMARK Instruction scheduling\ExPRESS - Benchmarks.htm

ملخص:

العمل التالي يعرض عملية تطبيق خوارزمية ***ماكس مين أونت سيستم*** المستوحاة من طريقة عيش مستعمرات النمل لحل مشكلة ترتيب التعليمات. والهدف ايجاد حل مقبول لمشكلة جدولة التعليمات في حضور الارتباطات والقيود المفروضة بين مختلف أجزائها. بعبارة أخرى إعادة ترتيب التعليمات من أجل تحسين وقت التنفيذ دون انتهاك قيود الأسبقية، وسنقدم مقارنة بينها وبين خوارزمية القوائم.

الكلمات المفتاح :

ترتيب المعلومات، مستعمرات النمل, ماكس مين أنت سيستم ،خوارزمية القوائم.

Abstract:

This work introduces the adaptation of MAX-MIN Ant System algorithm of Ant Colony to solve instruction scheduling.

The objective is to find an acceptable solution to the instruction scheduling problem in the presence of data dependences and resource constraints only, i.e., local instruction scheduling. In other words, ordering instructions within a basic block so that the schedule length is minimized without violating precedence constraints, (dependence constraints) together, between the two strategies the list algorithm (list scheduling) and MMAS.

Keywords: Instruction Scheduling, Ant Colony, MAX-MIN Ant System, List Scheduling,

Résumé:

Le travail suivant présente l'adaptation d'algorithme MAX-MIN Ant System de colonies de fourmis pour la résolution du problème de l'ordonnancement des instructions.

Le but est de trouver une solution acceptable au problème d'ordonnancement des instructions dans la présence des dépendances de données et des contraintes de ressources. En d'autres termes، réordonner les instructions d'un bloc de base dans le but d'en optimiser le temps d'exécution sans violer les contraintes de précédence, nous allons faire une étude comparative entre les deux stratégies l'algorithme de liste (liste scheduling) et MMAS.

Mots clés: l'ordonnancement des instructions, colonies de fourmis, MAX-MIN Ant System, les algorithmes de liste.