Replies to referee's comments on the paper "Using Knowledge-Based Systems for Research on Parallelizing Compilers" (Referee #1)

First of all, we wish to express our gratitude to the anonymous referee for his/her helpful suggestions and valuable comments. We have read the referee's comments very carefully, and modified our paper according the referee's comments. A summary of revisions is in the following.

1. **Reply to comment: Improve the english. It is quite good but still a bit quaint at places.**

   We have improved some grammatical structures to clearly express our technical contents accroding to referee's comments.

2. **Reply to comment: Focus the paper on the essential contribution "Knowledge-based System" and change title/oranization/abstract etc. accordingly. The other material can be kept as it has pedagogical value but viewed as "context".**

   We have reorganized Sections 2.1, 2.2 and 3. We moved Section 3.1 (old version) and 3.2 (old version) to Section 2.2 and 2.3, and moved Section 2.2 (old vserion) to Section 3.1. to focus the contribution on "knowledge-based approach." We also have revised the title/abstract/introductin/conclusion accroding to referee's comments.

3. **Reply to comment: Have a crisp section really evaluating value and potential of their approach as compared to other efforts in academia/commercial world.**

   We have added and enhanced the description about Section 4.4 (Section 4.4 is new version) by adding an description that compares our approach to other researches. In this paper, we concentrate on the fundamental phase, parallel loop scheduling, in parallelizing compilers running on multiprocessor systems. A new model exploiting loop parallelization using knowledge-based techniques is proposed. The knowledge-based approach integrates existing loop schedules to make good use of their abilities in extracting more parallelism. Experimental results show that the high speedup obtained by using IPLS on multiprocessors is obvious. Furthermore, for system maintenance and extensibility, our approach is obviously superior to others. In addition, a run-time technique based on the inspector-executor scheme is proposed to find available parallelism on loops. Our inspector can determine the wavefronts by building a DEF-USE table for each loop of a program. The process of the inspector for finding the wavefronts can be parallelized fully without any synchronization. Our executor can execute the loop iterations concurrently. Additionally, our compiler is highly modularized so that porting to other platforms is very easy, and it can partition parallel loops into multithreaded codes based on several loop-partitioning algorithms. The experimental results clearly show that the compiler achieves good speedup on Windows NT OS.