An Extended GLB Library for Optimization Problems

Shota Izumi, Daisuke Ishii
University of Fukui

Kazuki Yoshizoe
AIP, RIKEN

1. Search-based optimization

Optimization problems

- Obtain an assignment to the ***variables*** that minimizes the value of the ***objective function***, at the same time satisfying the ***constraints***
- Applicable to various practical problems
  *E.g. robotics, biology, economics
- Search is a basic but powerful method for optimization
  *Cf. nonlinear global optimization

Parallel-search-based optimization

- parallelizing optimizers to run efficiently on PC clusters is a promising approach

However……

most existing methods more or less assume a **centralized** structure
➡ scalability on massive PC clusters is limited

2. Summary of work

Extension of **the X10 GLB library** for parallel and distributed search computation

⬇

Provide a decentralized parallelization scheme for parallelizing various search-based optimization processes

Carefully designed benchmark for parallel-search-based optimization

⬇

Trial-and-error, tweaking, and performance-evaluation of the library through solving several instances of the benchmark

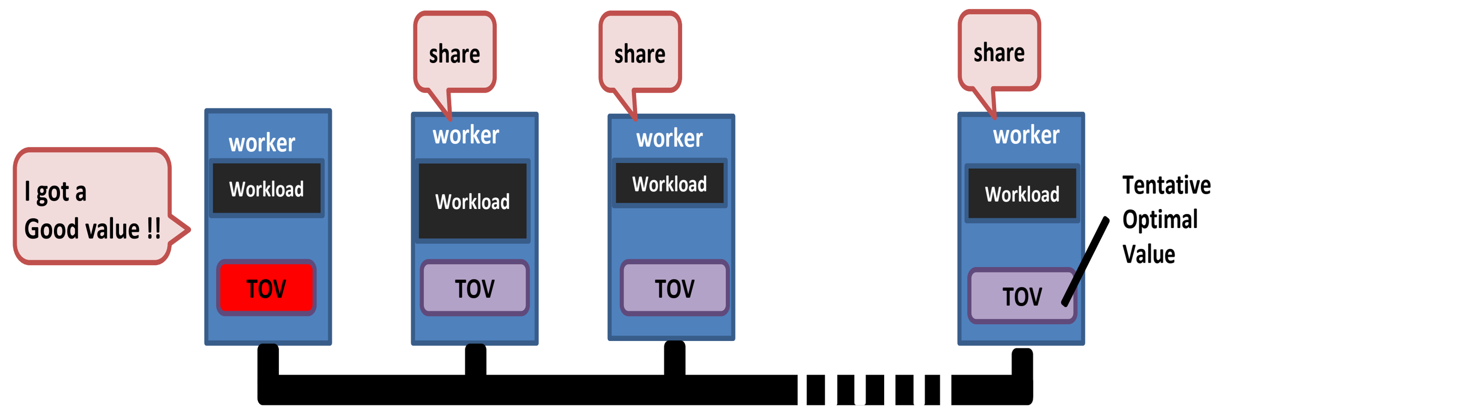
3. Extend GLB for Optimization Problem

X10 GLB (Global Load Balancing) Library [1]

- Effective parallelization scheme for non-uniform parallel tasks
- Performed by a decentralized homogeneous ***workers***
- Provides load-balancing and termination mechanisms

Our extension for optimization problems

- Distribution mechanism for locally-optimal feasible solutions
- A feasible solution may abandon a lot of workloads on other workers



[1] W. Zhang et al. 2014. GLB : Lifeline-based Global Load BalancingLibrary in X10. In PPAA. 31–40.

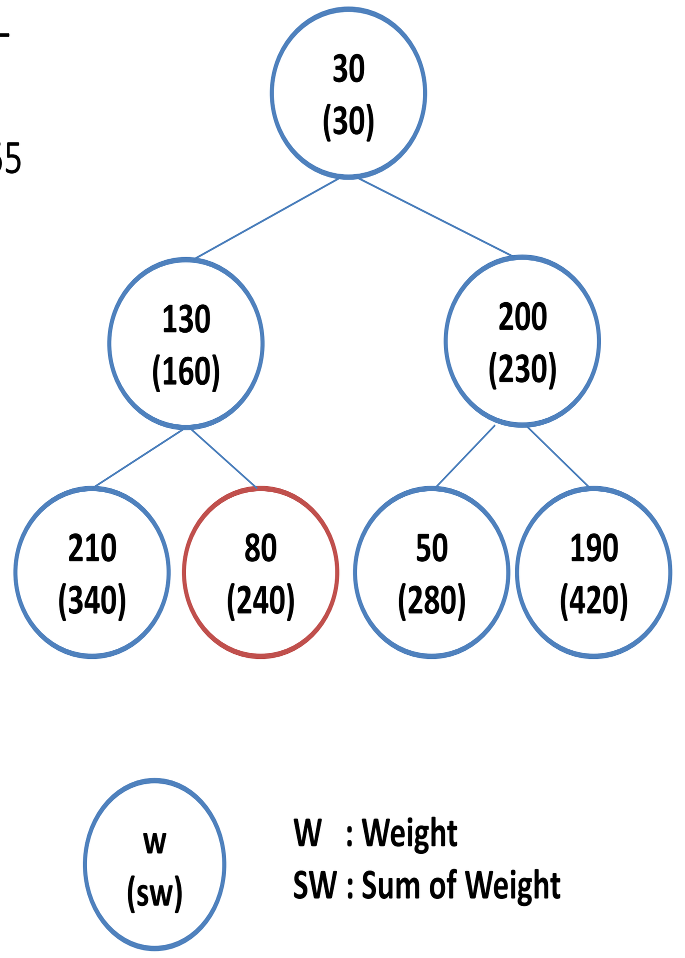
4. Benchmark

The designed problem is based on a perfect *b*-ary tree

- Nodes are weighted with random Integers 0-255
- Objective: find a path with a smallest sum weight

An instance (*b*, *d*, *h*)
b : the degree of branching
d : the depth of tree
h : the seed for random weight generation

An Extend : ranking top k feasible solutions
➡ More chances to have a tentative optimum
➡ Efficiency of distribution becomes crucial



5. Experiment

Two experiments to solve the benchmark with the GLB-based solver

Experiment environment (supercomputer of ACCMS, Kyoto University)

- Used 14 nodes
  - Each node has two Xeon 2.1GHz processors (18 × 2 cores, max. 504 cores in total) and 128GB RAM
- Used the C++ native compiler version 2.5.4 with MPI back-end.

First experiment

Evaluated the efficiency of the parallel solver using up to 288 cores

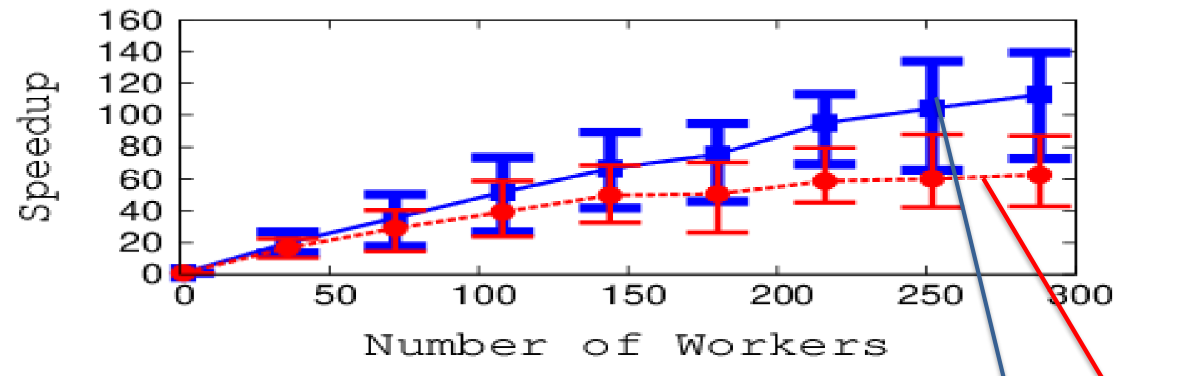
- In this experiment, tentative optima were ***broadcasted***
- Average for the instances with different hashes *h*
  – $h \in (0,13,15,25)$

Second experiment

Checked that the distribution of tentative optima with random sending improves the performance when the communication cost is a bottleneck

- We solved the instances in the first experiment using up to 504 cores (*K*=10)
- We used Two distribution methods
  - Send an optimum to all other workers (i.e. ***broadcasting***)
  - Send an optimum to randomly selected 1–2 workers (i.e. ***random sending***)

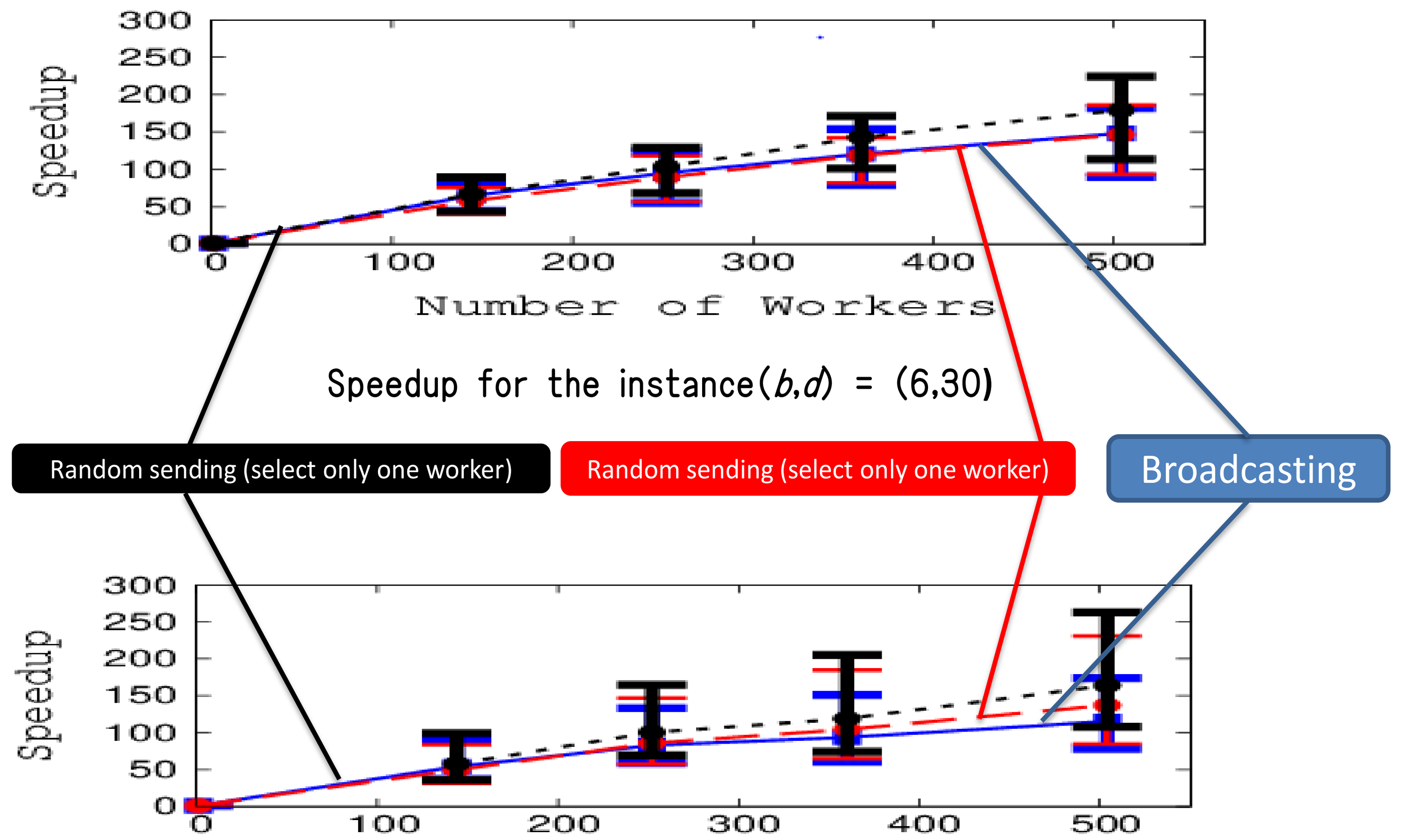
6. First experiment



Speedup for the instance(*b*,*d*) = (6,30)

Speedup for the instance(*b*,*d*) = (8,26)

7. Second experiment



Speedup for the instance(*b*,*d*) = (6,30)

Speedup for the instance(*b*,*d*) = (8,26)

8. Discussion

First experiment

- We had monotonic speedups despite broadcasting the optima
- Range of the speedups for each instance group was reasonably small

Second experiment

- Parallelization scheme scales up to 504 cores
- Random stealing improves the distribution efficiency

Benchmark problem

- Expected to help further development of the GLB library
- Controllability of # of feasible solutions, possibility of search space pruning, etc.

conclusion

Preliminary but promising report on parallelizing search-based optimization processes

- Combined global load balancing and information sharing mechanisms