Reliable High Performance Computing Strategies for Chemical Process Modeling

Chao-Yang Gau and Mark A. Stadtherr[1]
Department of Chemical Engineering
University of Notre Dame
Notre Dame, IN 46556 USA

AIChE Annual Meeting, Dallas, TX, Oct. 31– Nov. 5, 1999
Session Number 10004: High Performance Computing

[1]Author to whom all correspondence should be addressed. Phone: (219)631-9318; Fax: (219)631-8366; E-mail: markst@nd.edu

Outline

- Interval Analysis
  - Interval Newton/Generalized Bisection (IN/GB).
  - Reliability in finding (enclosing) all solutions and in solving global optimization problems.
- Improvements in underlying algorithm of IN/GB:
  - Row-wise construction of optimal preconditioners.
  - Application of hybrid pivoting preconditioner rows.
- Improvements in parallel computing strategy:
  - Asynchronous Diffusive Load Balancing (ADLB) for overlapping computation and communication.
  - Scalable parallel virtual network.
  - Effective parallel branch-and-bound in global optimization.
- Experiments and Results
- Concluding Remarks

Summary

- Interval analysis combined with parallel computing provide the necessary computational power for realistic and reliable process simulation and optimization.

- Enhancement in interval analysis:
  - The use of hybrid pivoting preconditioner has led to large reductions in CPU time for all problems tested, and provides the capacity to efficiently solve larger problems.

- Enhancement in parallel computing:
  - ADLB on 2-D torus virtual network has achieved high scalability with efficiency at 92% .
  - Novel parallel branch-and-bound has resulted in more consistent high superlinear speedups for global optimization on up to 32 processors.

- Parallel computing offers can provide advantages which are not available by serial computing.

Background—Interval Analysis

- A real interval $X = [a, b] = \{x \in \Re \mid a \leq x \leq b\}$ is a segment on the real number line and an interval vector $\mathbf{X} = (X_1, X_2, ..., X_n)^T$ is an n-dimensional rectangle or "box".

- Basic interval arithmetic for $X = [a, b]$ and $Y = [c, d]$ is X op $Y = \{x \text{ op } y \mid x \in X,\ y \in Y\}$ where op $\in \{+, -, \times, \div\}$. For example, $X + Y = [a + c, b + d]$.

- Computed endpoints are **rounded out** to guarantee the enclosure.

- Interval elementary functions (e.g. $\exp(X)$, $\log(X)$, etc.) are also available.

- The interval extension $F(\mathbf{X})$ encloses all values of $f(\mathbf{x})$ for $\mathbf{x} \in \mathbf{X}$. That is, $F(\mathbf{X}) \supseteq \{f(\mathbf{x}) \mid \mathbf{x} \in \mathbf{X}\}$.

- Interval extensions can be computed using interval arithmetic (the "natural" interval extension), or with other techniques.

Interval Newton Method

- For the system of nonlinear equations $\mathbf{f}(\boldsymbol{x}) = \mathbf{0}$, find (enclose) **with mathematical and computational certainty** all roots in a given initial interval $\mathbf{X}^{(0)}$ or determine that there are none.

- At iteration k, given the interval $\mathbf{X}^{(k)}$, if $0 \in \mathbf{F}(\mathbf{X}^{(k)})$ solve the linear interval equation system

$$F'(\mathbf{X}^{(k)})(\mathbf{N}^{(k)} - \boldsymbol{x}^{(k)}) = -\mathbf{f}(\boldsymbol{x}^{(k)})$$

  for the "image" $\mathbf{N}^{(k)}$, where $\mathbf{F}(\mathbf{X}^{(k)})$ is an interval extension of $\mathbf{f}(\boldsymbol{x})$ and $F'(\mathbf{X}^{(k)})$ an interval extension of its Jacobian over the current interval $\mathbf{X}^{(k)}$, and $\boldsymbol{x}^{(k)}$ is a point inside $\mathbf{X}^{(k)}$, often the midpoint $\mathbf{m}^{(k)}$.

- Any root $\boldsymbol{x}^* \in \mathbf{X}^{(k)}$ is also contained in the image $\mathbf{N}^{(k)}$, suggesting the iteration scheme $\mathbf{X}^{(k+1)} = \mathbf{X}^{(k)} \cap \mathbf{N}^{(k)}$ (Moore, 1966).

- Interval Newton also provides an existence and uniqueness test:

Interval Newton Method (continued)

- *True*: If $\mathbf{N}^{(k)} \subset \mathbf{X}^{(k)}$, then there is a **unique** zero of $\mathbf{f}(\boldsymbol{x})$ in $\mathbf{X}^{(k)}$, and the point Newton method will converge quadratically to the root starting from any point in $\mathbf{X}^{(k)}$.

- *False*: If $\mathbf{X}^{(k)} \cap \mathbf{N}^{(k)} = \emptyset$ or $0 \notin \mathbf{F}(\mathbf{X}^{(k)})$, then there is no root in $\mathbf{X}^{(k)}$.

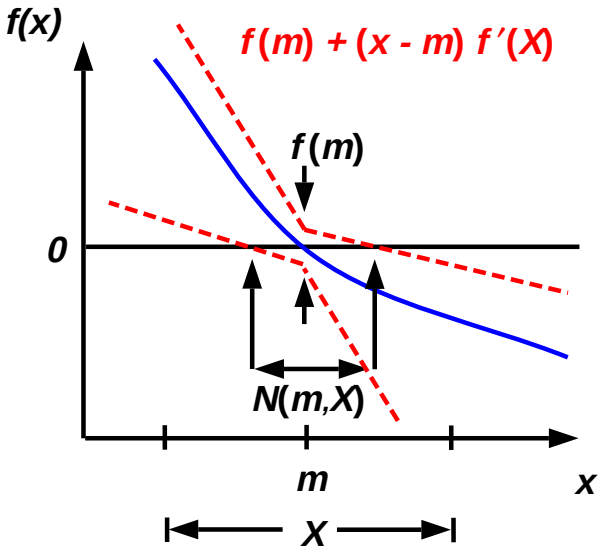
- *Unknown*: Otherwise, then either:
  - Continue with the next iterate $\mathbf{X}^{(k+1)}$ if it is sufficiently smaller than $\mathbf{N}^{(k)}$, or
  - **Bisect** $\mathbf{X}^{(k+1)}$ and perform interval Newton on the resulting intervals.

  This is the interval Newton/generalized bisection (IN/GB) approach.

- Basically, it follows a **branch-and-prune** scheme :
  - If test is true or false, then prune node.
  - If test is unknown and bisect, then branch (bisect node), generating a binary tree structure.

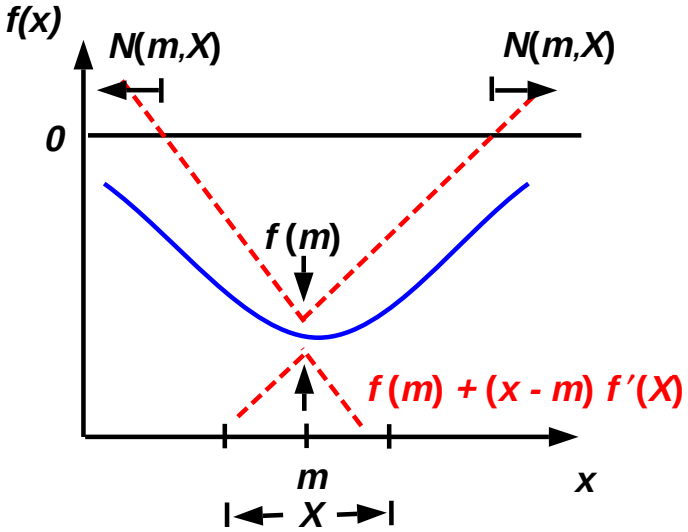
$N(m,X) \subset X \Rightarrow$

Proof of existence



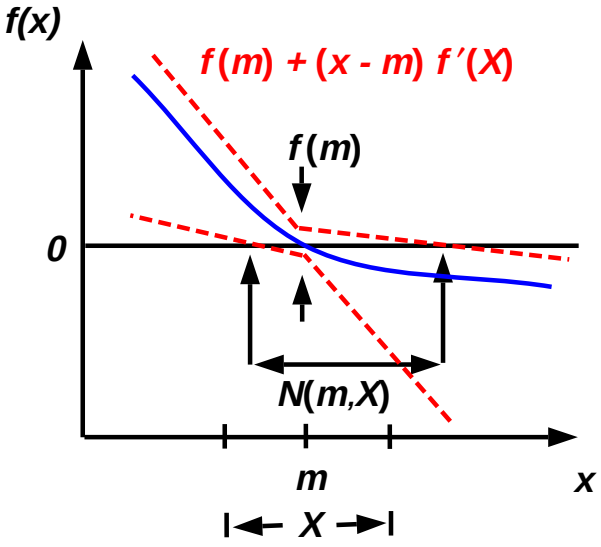
$N(m,X) \cap X = \varnothing \Rightarrow$

Proof of non-existence



$N(m,X) \not\subset X \Rightarrow$

$X' = N(m, X) \cap X$



Preconditioned Interval Gauss-Seidel

- Preconditioner in interval Gauss-Seidel Scheme:
  - A matrix $Y^{(k)}$ which takes inner product with interval Jacobian matrix and real function vector.

$$Y^{(k)}\mathbf{F}'(\mathbf{X}^{(k)})(\mathbf{N}^{(k)} - \boldsymbol{x}^{(k)}) = -Y^{(k)}\mathbf{f}(\boldsymbol{x}^{(k)})$$

  - Stabilizes Gauss-Seidel iteration and leads to tighten enclosure of solution.
  - Often chosen to be inverse of midpoint interval Jacobian matrix or inverse of Jacobian matrix at midpoint - inverse midpoint preconditioner Y^{inv}.

- Effectiveness of preconditioners dominates the performance of Gauss-Seidel Scheme given tighten bounds on functions.

- Optimal Preconditioner: effectively shrink width for interval (box) of interest and efficiently remove an interval (box) out of interest.

Row-wise View of Preconditioners

- Construct a preconditioner row by row in the progress of each component iteration.

- Consider the i-th coordinate of Gauss-Seidel iteration and the i-th preconditioner row, Y_i,

$$\begin{aligned} \mathbf{N_i} &= x_i - \frac{\mathbf{Q_i}(\mathbf{y_i})}{\mathbf{D_i}(\mathbf{y_i})} \\ &= x_i - \frac{\mathbf{y_i f}(\mathbf{x}) + \sum_{\substack{j=1 \\ j \neq i}}^{n} \mathbf{y_i A_j}(X_j - x_j)}{\mathbf{y_i A_i}}, \end{aligned}$$

  then take $(\mathbf{X_i} \cap \mathbf{N_i})$.

  $\mathbf{A_i}$ is the i-th row of $\mathbf{F}'(\mathbf{X})$ matrix.

- C-Preconditioner (Y^C): *contract* intervals, provided $0 \notin \mathbf{D_i}$, and end up a single connected interval $\mathbf{N_i}$.

- E-Preconditioner (Y^E): *split* intervals, provided $0 \in \mathbf{D_i}$, $0 \neq \mathbf{D_i}$ and $0 \notin \mathbf{Q_i}$, and end up with an extended interval $\mathbf{N_i}$, consisted of two semi-infinite intervals.

Optimality Criteria for Precond. Row

- Practical optimality criteria:
  - Width-optimal preconditioner row : minimize width of $(\mathbf{X_i} \cap \mathbf{N_i})$.
  - Endpoint-optimal preconditioner row: maximize the lower bound of $\mathbf{N_i}$ or minimize the upper bound of $\mathbf{N_i}$.

- Approached by hybrid scheme with pivoting preconditioner row, which contains only one non-zero element equal 1.

- Manipulate element Y_{ij} for $j = 1, ..., n$ to be 1 and look for the optimal image corresponding to the i-th coordinate:

$$\begin{aligned}(\mathbf{N_i})_j &= x_i^m - (\frac{\mathbf{Q_i}}{\mathbf{D_i}})_j \\ &= x_i^m - \frac{f_j + \sum_{\substack{k=1 \\ k \neq i}}^{n} A_{jk}(X_k - x_k)}{A_{ji}}\end{aligned}$$

Hybrid Pivoting Scheme

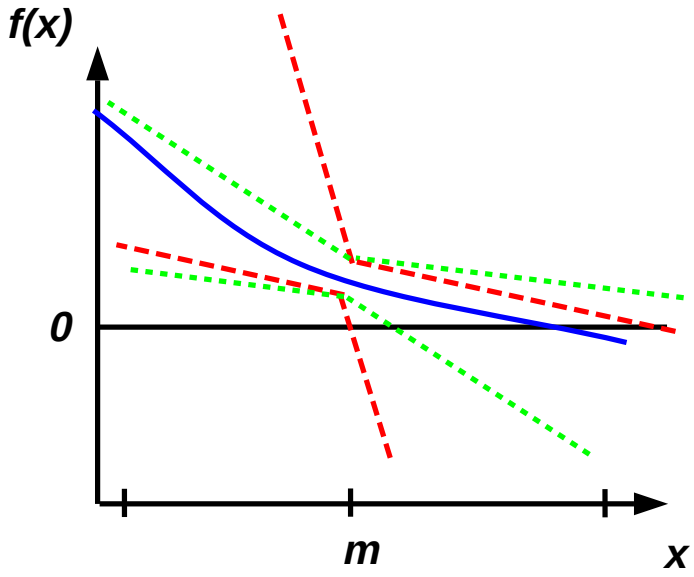
- Take advantages of pivoting preconditioner row, Y_i^P, through a hybrid scheme for removing or mostly shrinking a box.

- Compare each $(\mathbf{X_i} \cap (\mathbf{N_i})_j)$ over $j = 1, .., n$, determine which y_{ij} to manipulate or be non-zero for optimality criteria:
  - If $\mathbf{X_i} \cap (\mathbf{N_i})_j = \varnothing$, can use to remove the box,
  - Otherwise, could use to minimize $w(\mathbf{X_i} \cap \mathbf{N_i})$.

- Hybrid scheme for incorporation of Y^{inv} and Y^P:
  - Determine optimal Y_i^P and estimate $(\mathbf{X_i} \cap (\mathbf{N_i})_j)$ in rough interval arithmetic.
  - If predict empty $(\mathbf{X_i} \cap (\mathbf{N_i})_j)$, use Y_i^P in Gauss-Seidel step.
  - Otherwise, use Y_i^{inv} first. Compare the width of resulting box and $w(\mathbf{X_i} \cap (\mathbf{N_i})_j)$, then decide whether or not to re-do Gauss-Seidel step using Y_i^P.

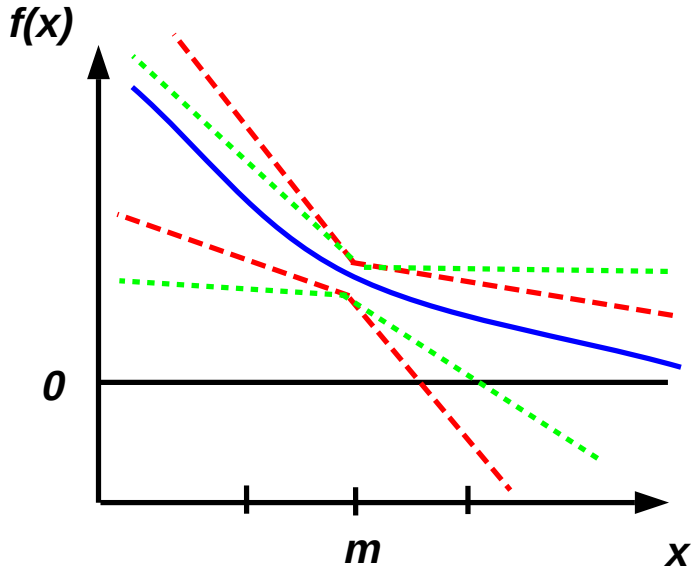
Pivoting C-Preconditioner

$0 \in X$



$N \cap X$

$0 \notin X$

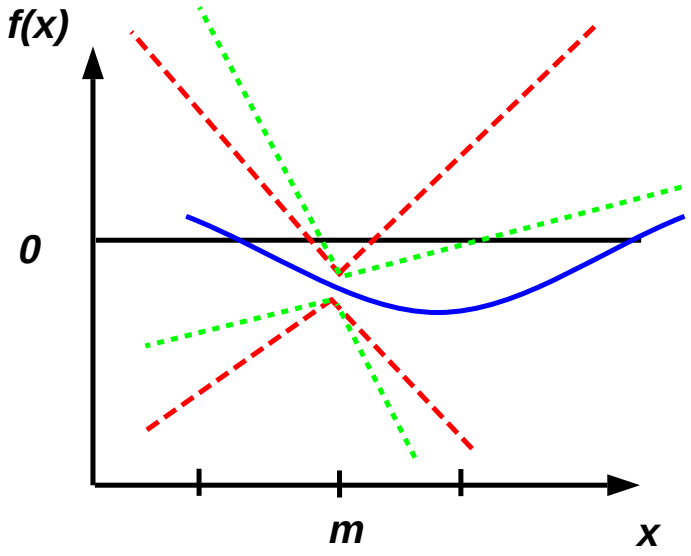


$N \cap X$

$\varnothing$

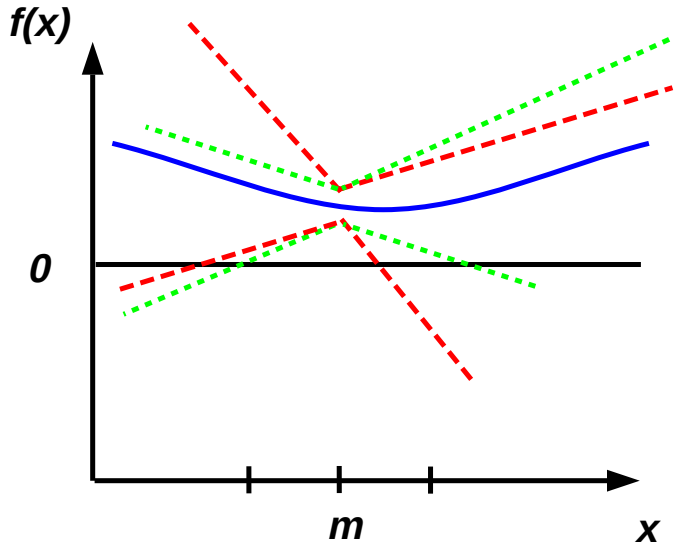
Pivoting E-Preconditioner

$0 \in X$



$N \cap X$

$0 \notin X$



$N \cap X$

$\varnothing$

Numerical Tests and Results

- Both equation-solving and global optimization problems were selected to illustrate improvements.

- Compare computational cost for Sun Ultra 2/1300 with the use of different preconditioners.

- Prob. 1: phase stability analysis [Tessier, 1997]
  - Equation-solving problem with 6 component variables.
  - Use UNIQUAC activity coefficient model to compute Gibbs energy in LLE systems.
  - Using Y^{inv} took 50217 CPU seconds
  - Using hybrid precond. took 152 CPU seconds.

- Prob. 2:computation of critical points [Stradi, 1999]
  - Equation-solving problem with 6 variables for 4 components.
  - Use Peng-Robinson equation of state to model both the liquid and gas phases in mixtures.
  - Using Y^{inv} took 2094 CPU seconds
  - Using hybrid precond. took 658 CPU seconds

Numerical Test and Results (Cont.)

- Prob. 3: computation of heterogeneous azeotropes [Maier, 1999]
  - Equation-solving problem with 11 variables for 3 components.
  - Use NRTL activity coefficient model for equal fugacity condition.
  - Using Y^{inv} took $>$ 1 CPU days
  - Using hybrid precond. took 270 CPU seconds.

- Prob. 4: error-in-variable parameter estimation [Kim, 1990; Esposito, 1998]
  - Global optimization with 2 parameter variables and 10 state variables.
  - Point evaluations of object function done at the midpoint of current box is used for objective range test.
  - Use Van Laar equation to model experimental vapor-liquid equilibrium data.
  - Using Y^{inv} took $>$ 4 CPU days
  - Using hybrid precond. took 1504 CPU seconds.

Concluding Remarks for Improvements in IN/GB

- Hybrid pivoting preconditioner scheme provides an approach to manipulate interval Gauss-Seidel process.

- The use of hybrid pivoting preconditioner has led to large reductions in CPU time for all problems tested, and in some cases, reductions of 2 or more orders of magnitude.

- Highly difficult problems benefit from more work in each iteration to eliminate redundant interval evaluations.

Parallel Computing and IN/GB

- Take advantage of rapid growth in computational power of parallel processing systems.

- Network-connected workstations
  - Economical
  - Universally available
  - Reliable

- Multiple processors can be used to concurrently perform IN/GB in disjoint parts (intervals) of the binary tree.

- The binary trees may be highly irregular, and can result in highly uneven distribution of work among processors and thus poor overall performance.

- Apply an effective load scheduling and load balancing scheme to do parallel tree search efficiently.

Asynchronous Diffusive Load Balancing

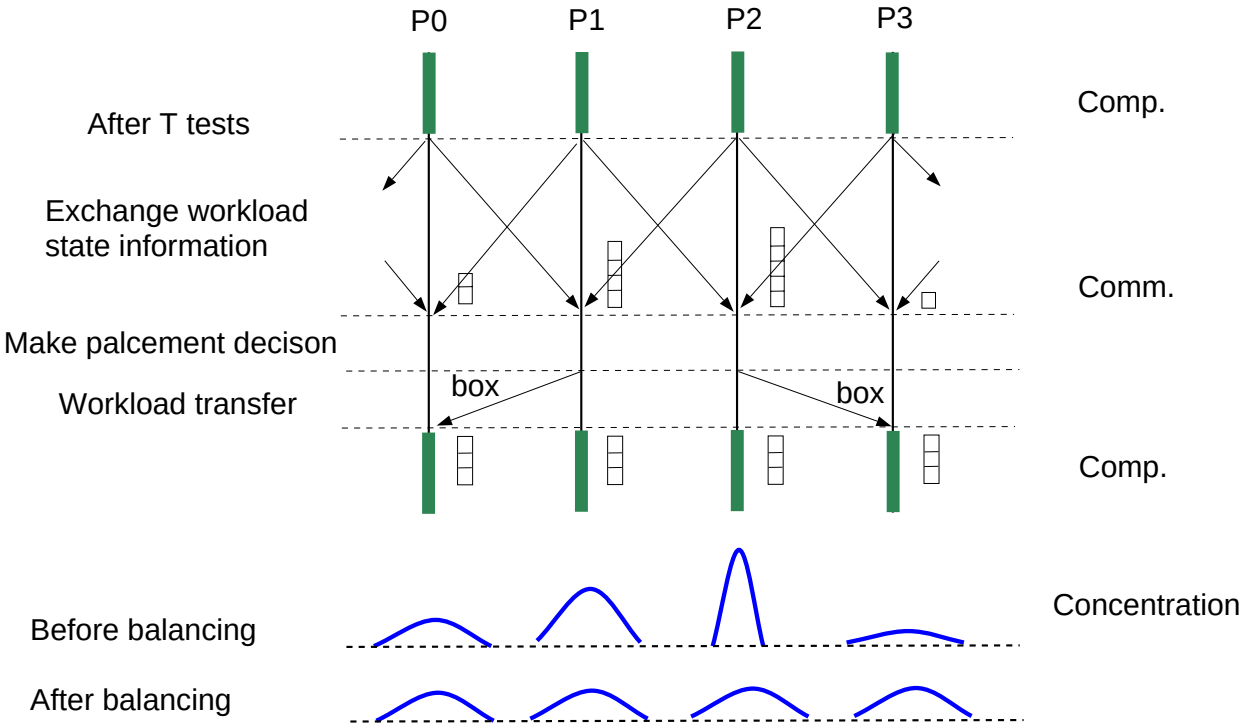
- Irregular parallel search trees need load balancing to redistribute workload concurrently at runtime.

- Distributed System: coordinate all processors to maintain distributed interval stacks and prevent idle states through workload transfer over network.

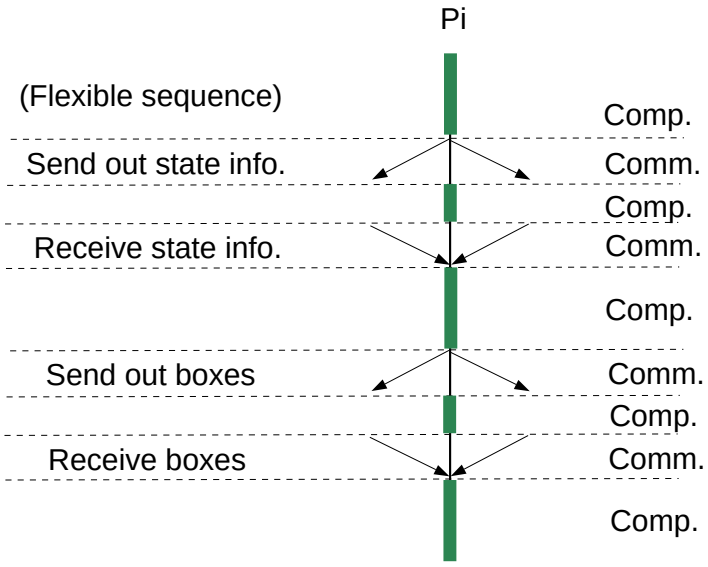
- Asynchronous Diffusive Load Balancing (ADLB) [Gau and Stadtherr, 1998]
  - Use asynchronous nonblocking communication and persistent communication to update workload information and transfer workload.
  - Overlap communication and computation and reduce appearance of idle state.
  - Receiver Initiate and Local Communication: Exchange workload information with their immediate neighbors. Once local workload less than certain threshold, request neighbors for stack boxes.

Diffusive Load Balancing Scheme

- **Synchronous**

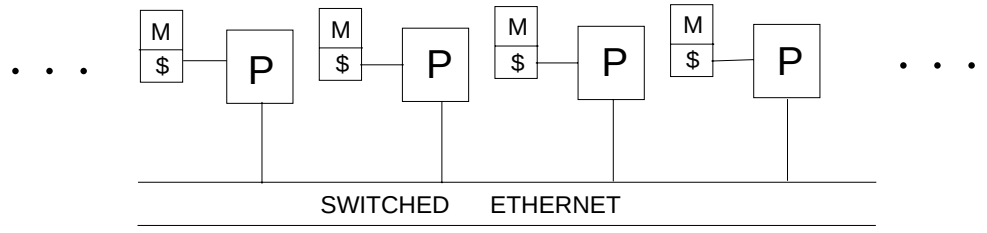


- **Asynchronous**



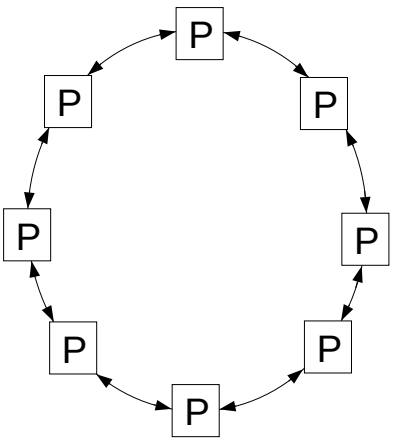
Parallel System

- Physical Architecture:
  Sun Ultra Enterprise 2 workstations connected by switched Ethernet

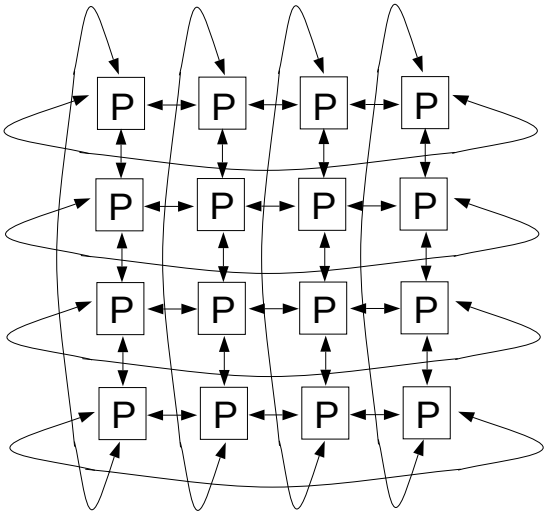


- Virtual Algorithm
  - Software: implemented in Fortran 77 using message-passing interface (MPI) protocol
  - Virtual Network: local communication

1-D Torus Network



2-D Torus Network



Parallel Branch-and-Prune for Equation-Solving Problem

- Local computation phase: IN/GB
- Global communication phase: ADLB - maintain distributed workload (box) stack
- Distributed termination detection: modified Dijkstra's token algorithm
  - Proved to be a reliable solution to termination effects in distributed system.
  - Less idle time waiting for termination.
- Performance Enhancement
  - Appropriately organize communication sequence.
  - Optimize program structure for communication and computation.
  - Attempt to minimize isoefficiency function.

Parallel Branch-and-Bound for Global Optimization

- A node can also be pruned earlier if the interval extension $\boldsymbol{\Phi}(\boldsymbol{X})$ of the objective $\phi(\boldsymbol{x})$ has a lower bound greater than the current best (least) upper bound. This follows a **branch-and-bound** scheme.

- Best upper bound is determined and updated by:
  - Upper bound of $\boldsymbol{\Phi}(\boldsymbol{X})$, and/or
  - Point function evaluations with interval arithmetic in each interval tested, and/or
  - Running a local optimizer.

- Once a better upper bound is generated, the value is diffusively broadcast over all processors.

- The way an interval box is transfered to and is examined by various processors is determined dynamically in parallel processing.

- The update of best upper bound in sequence and in parallel is performed in different manner, and can be considered as a $nondeterministic$ process.

Dual Stack Management

- Using $random\ process$ to manage nondeterministic box-processing might alleviate irregularity of parallel search.

- Split each box stack into two sub-stacks.

- Local Ordered Stack:
  - In charge of maintaining boxes for local computation (IN/GB).
  - Explore binary tree in $depth\text{-}first$ search.
  - Boxes are sequentially ordered.
  - Once a best upper bound is found, assign contiguous boxes to local stack to intensify bounding scheme.

- Global Random Stack:
  - Box storage for dynamic load balancing.
  - Contribute tree search with $breadth\text{-}first$ manner.
  - Boxes are $randomly$ selected for transfer.
  - Primarily comprised of boxes near the tree root.

Performance Metrics

- Scalability Analysis for equation-solving problems
  - Isoefficiency function [Kumar, 1994]
    Relate problem size to processor number necessary for increase in speedup in proportion to processor number.
  - Small isoefficiency function represents highly scalable parallel algorithm.
  - Applications: select best parallel algorithm and predict performance of specific parallel algorithm.

- Speedup Anomaly Analysis for global optimization
  - Search Overhead (SO) factor: the ratio of work done by parallel formulation to work done by sequential formulation
  - SO > 1 : deceleration anomaly and might result in sublinear speedup
    SO < 1 : acceleration anomaly and might result in superlinear speedup
  - An optimal parallel search will concentrate in acceleration anomaly.

Experiments and Results for Parallel Branch-and-Prune

- Parallel algorithms with 1-D and 2-D virtual connectivity, respectively, were examined by scalability analysis.

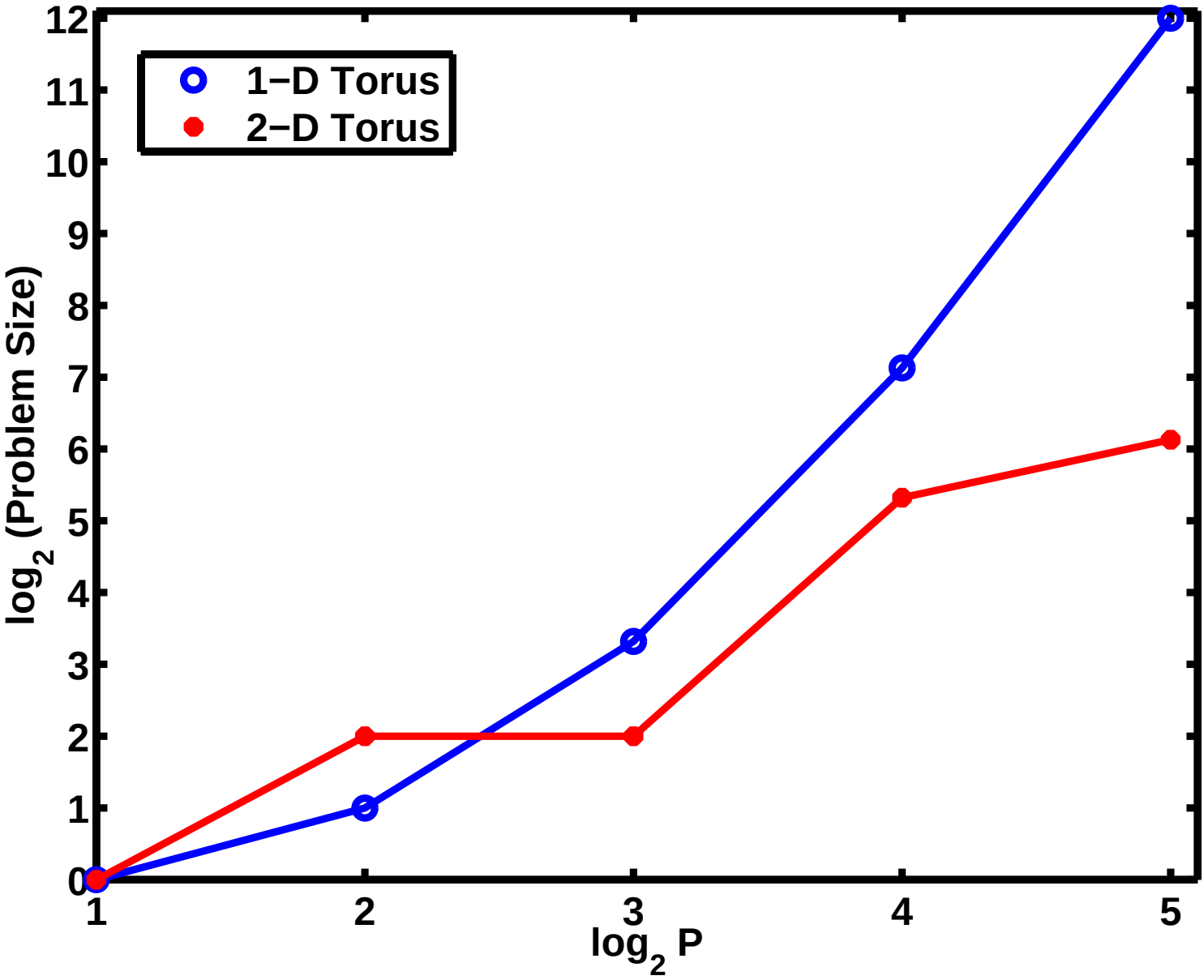
- An equation-solving problem, computation of critical points [Stradi,1999], is selected as the test example.

- Work size was increased stepwisely by assigning multiple times of identical initial intervals to an underlying instance.

- For constant efficiency at 92%, 2-D torus averagely has smaller isoefficiency function, thus more scalable than 1-D torus.

- On four processors, 1-D torus has an advantage with lower communication cost over 2-D torus. However, for a larger number of processors this advantage is overweighted by poor message distribution

Isoefficiency Curve for Equation-Solving Problems



Experiments and Results for Parallel Branch-and-Bound

- The effect on speedup anomaly was investigated when using single-stack management as well as dual-stack managements over 2-D torus.

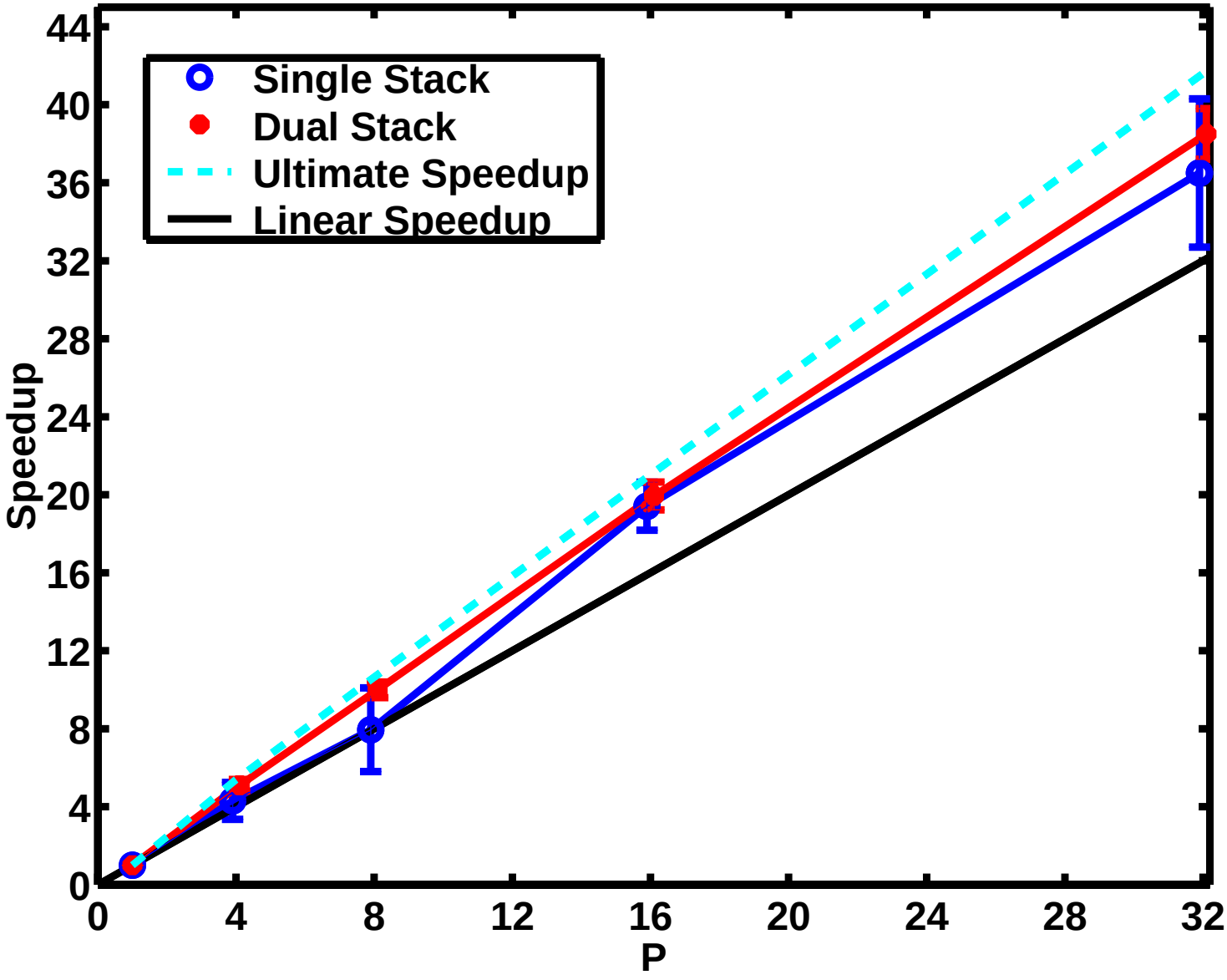
- An optimization type of problem, error-in-variable parameter estimation [Kim,1990; Esposito,1998], was used for testing.

- Ultimate speedups, which provide speedup upper bounds, can be obtained by initially setting best upper bound at globally minimal objective value.

- Experiments show that dual-stack management results in higher as well as more consistent speedups, and also prevents deceleration anomaly.

- Random process, which reduces virtual granuality, indeed alleviates irregularity of tree search on small number processors.

Speedup Anomaly for Optimization Problems



Concluding Remarks for Improvements in Parallel Computing

- We have developed a parallel branch-and-prune (and/or bound) scheme embedded in IN/GB for equation-solving and global optimization problems.
  - High efficiency in branch-and-prune scheme
  - Good speedup in branch-and-bound scheme.

- We presented two performance metrics for parallel algorithm: isoefficiency function and speedup anomaly.

- In terms of underlying parallelism, 2-D torus as virtual connectivity has higher scalability.

- In terms of parallel tree search, dual-stack management achieves higher search efficiency on multiprocessor system.

- Parallel algorithms can provide some advantages which are not available by sequential algorithms.

- Acknowledgments
  - ACS PRF 30421-AC9
  - NSF CTS95-22835, DMI96-96110 and EEC97-00537-CRCD
  - EPA R824731-01-0 and R826734-01-0
  - DOE DE-FG07-96ER14691
  - US Army Research Office DAAG55-98-1-0091

- For more information:
  - Contact Prof. Stadtherr at markst@nd.edu
  - Copies of slides will be available next week at http://www.nd.edu/~markst/presentations.html