Development of a Super-fast Numerical Equation Solver in Java

Tyler S. Helmus, Caleb W. Reese, Dr. Matthew Kuperus Heun

Department of Engineering, Calvin College, Grand Rapids, MI 49546

Objectives

There were two main objectives to our summer research:

- Develop a super-fast numerical equation solver in Java for large systems of equations
- Examine the speed advantages of utilizing parallel processing to solve these large systems of equations

Methods

Occurrence Matrices

- Occurrence matrices were used to model the equation sets by setting the value to true (1) for unknown variables in each equation

E0: $a + b = 7$

E1: $b - a = 3$

E2: $c = a + d$

E3: $d = bc$

E4: $f = a/b$

E5: $g^2 = b^2 - a$

E6: $ea = f^b$

e\v	a	b	c	d	e	f	g	ω(e)
E0	1	1	0	0	0	0	0	2
E1	1	1	0	0	0	0	0	2
E2	1	0	1	1	0	0	0	3
E3	0	1	1	1	0	0	0	3
E4	1	1	0	0	0	1	0	3
E5	1	1	0	0	0	0	1	3
E6	1	1	0	0	1	1	0	4
ω(v)	6	6	2	2	1	2	1	

Figure 1. Occurrence matrix example showing a simple system of equations with b = .29 and l = 0.

Methods (cont'd)

Blocking Algorithm

- The goal of this is to create a process which separates the equations into smaller blocks which can be solved independently when solved in a specific order. It outputs the blocks and the order.

Subsetting Algorithm within Blocking Algorithm

- The subsetting algorithm intelligently determines subsets of equations based on the variables. These can then be placed into the combination generator in order to find the blocks. The following graph shows the advantage of this method over a combination generator alone.

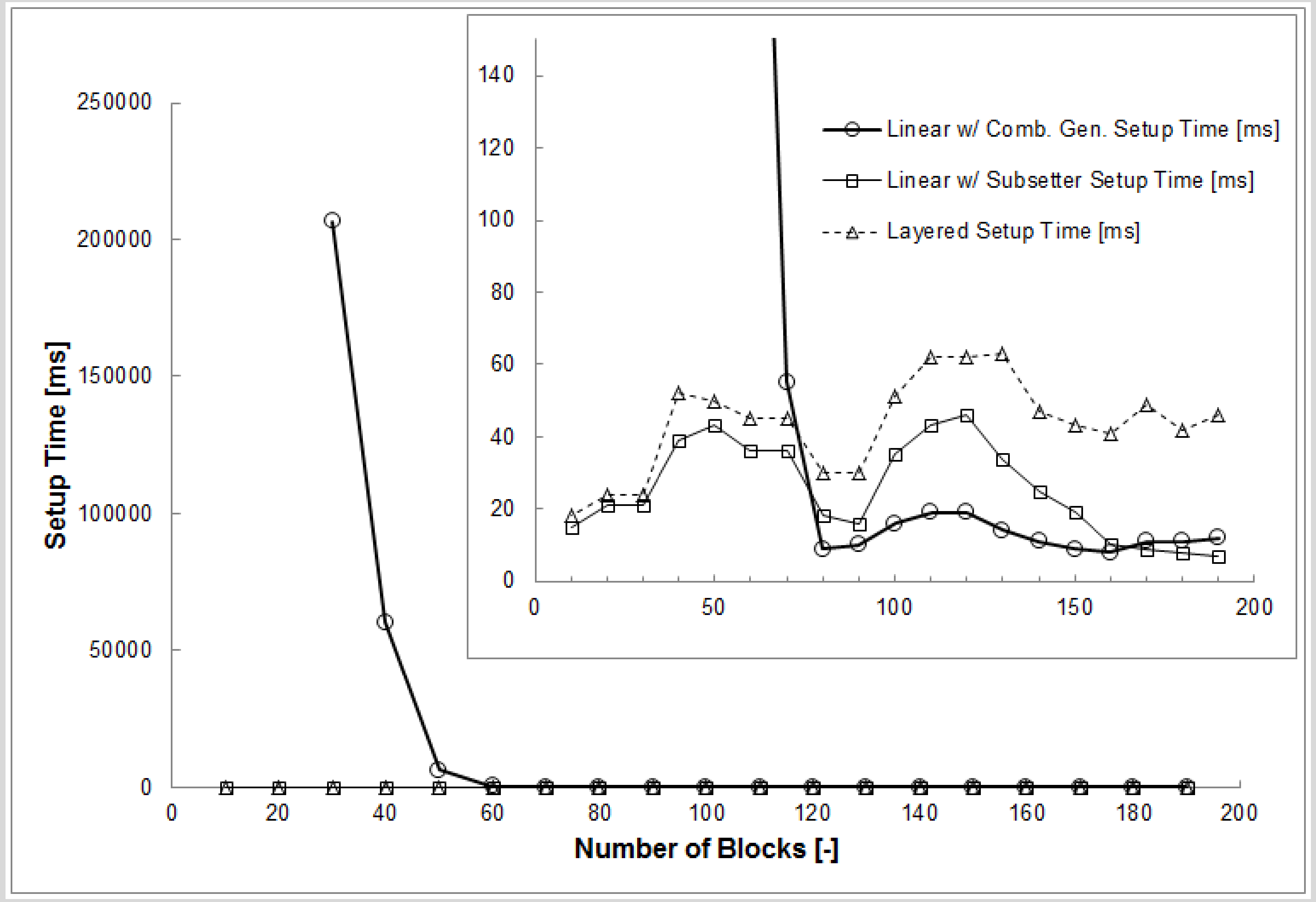


Figure 2. Justification of the Subsetting algorithm.

Methods (cont'd)

Layering Algorithm

- The layering algorithm uses the ordered output from the blocking algorithm and determines which blocks can be solved simultaneously on the same "layer," based on shared variables between blocks and the solution order.

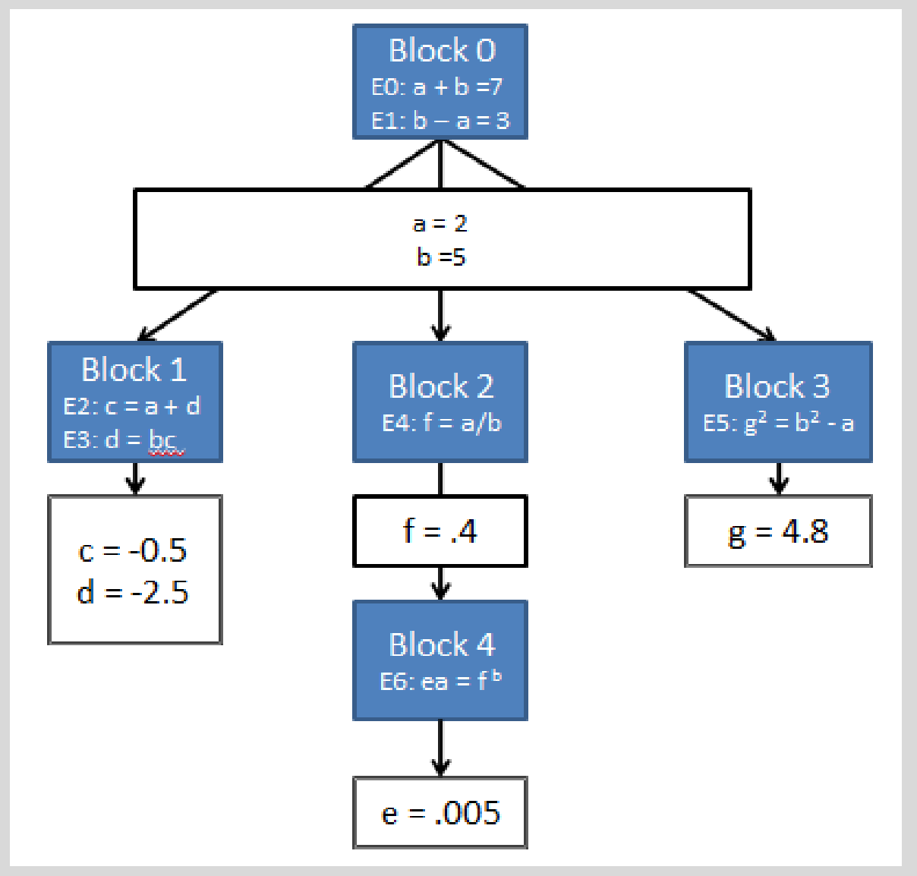


Figure 3. Equation set with layering. Blocks 1, 2, and 3 are on the same layer because they don't rely on the other blocks in the layer to be solved.

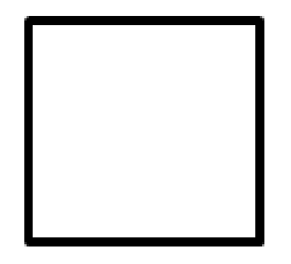
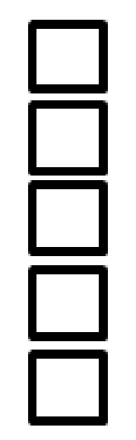
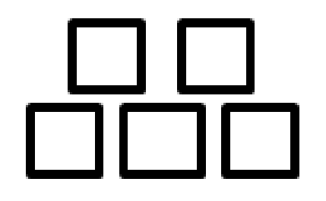
	Single Step	Linear	Layered
	 N = 100, B = 1, b = undefined, L = 1, l = undefined	 N = 100, B = 5, b = .04, L = B, l = 0	 N = 100, B = 5, b = .04, L = 2, l = .75
Setup Considerations	No Setup Time	Linear Algorithm Required	Linear and Layering Algorithms Required
Solve Considerations	$t_{Solve}:O(N^2)$	$t_{Solve}:O\left(B\left(\frac{N}{B}\right)^2\right)$ All blocks solved on same thread	$t_{Solve}:O\left(L\left(\frac{B/L}{N_{processors}}\right)\left(\frac{N}{B}\right)^2\right)$ Multiple threads will decrease t_{Solve}, overhead of thread management will increase t_{Solve}

Figure 4. Comparison of the different solution methods theoretical solve times and setup considerations.

Use your smartphone and the QR code to see the full paper or enter this web address: bit.ly/SolverPaper

CALVIN
Minds in the Making

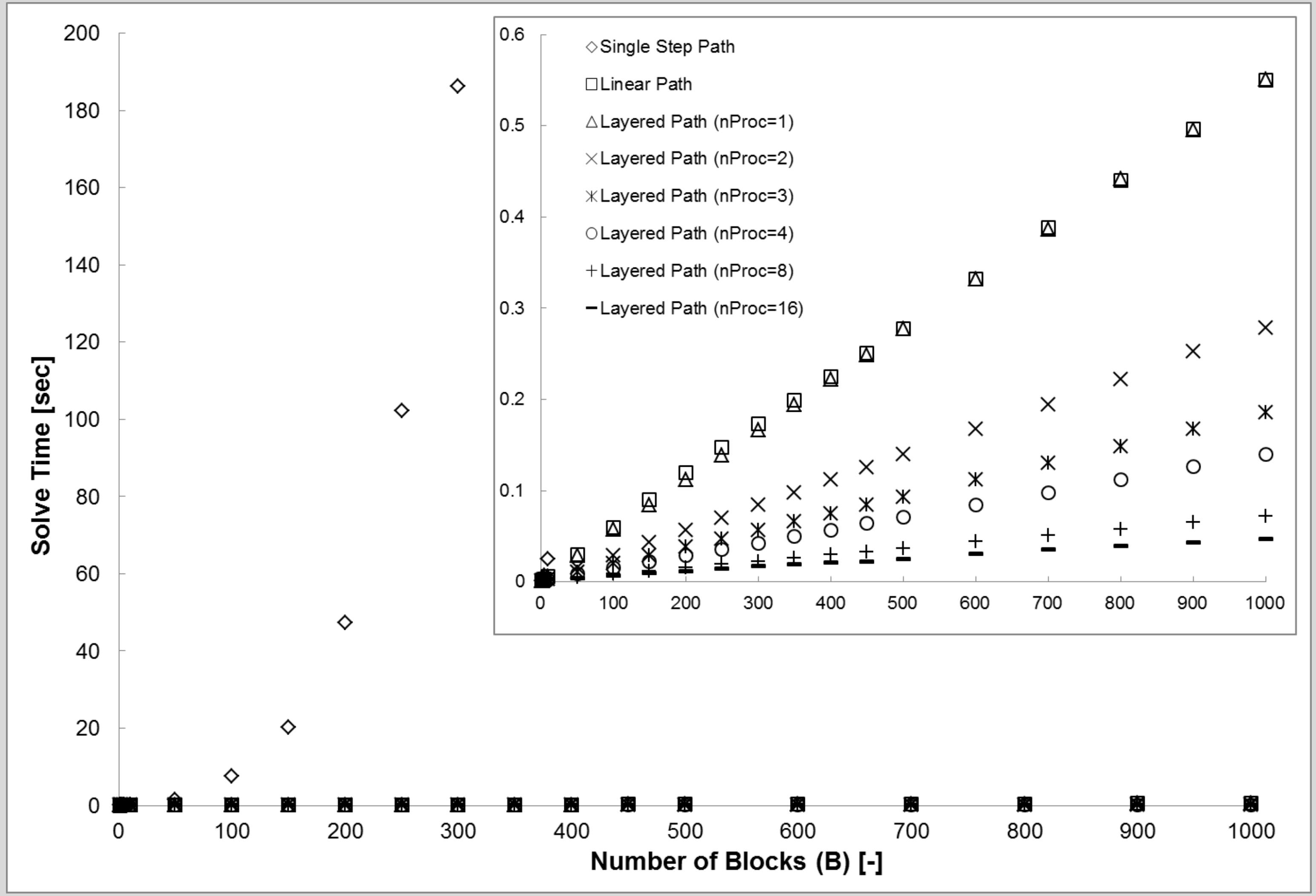


Figure 5. Solve time as a function of number of blocks (for multiple processors).

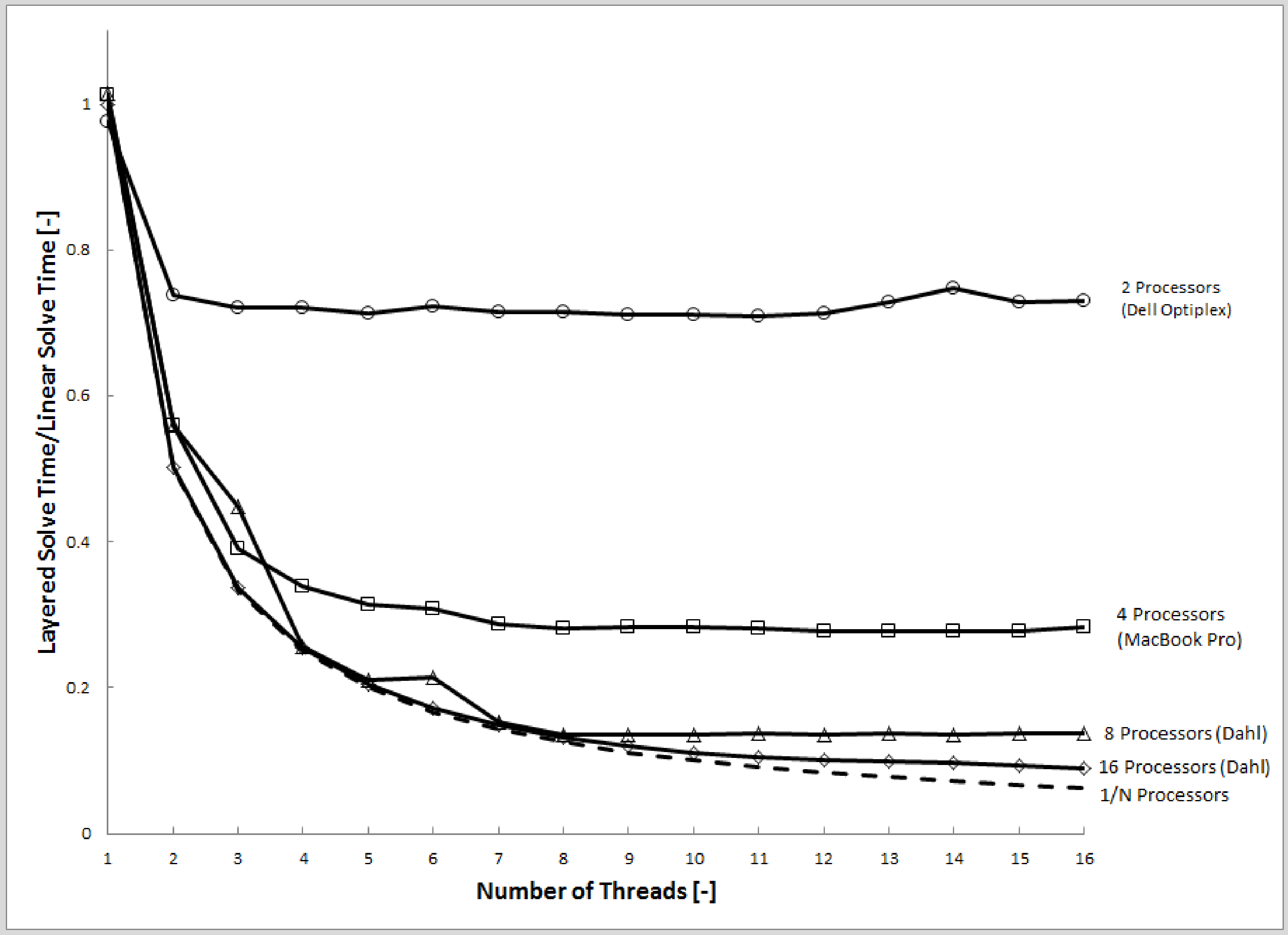


Figure 6. Relative solve times (Layered/Linear time) as a function of number of threads being used (N = 2000).

Conclusion

The performance of our algorithms depends on many factors, including the structure of the equation set, the number of available processors, and the overhead of thread management. With 2000 equations in 500 blocks with 16 processors, we show that the multi-core approach decreases the solve time by a factor of 11.2, or 70% of the theoretical speed increase, compared to single-core techniques.

References

Alexander, D. G., and D. M. Blackketter. "Optimized Solution Strategy for Solving Systems of Equations." *Engineering Computations* 20.2 (2003): 178-91. Print.