Introduction to Parallel Computing

Ananth Grama, Anshul Gupta, George Karypis, and Vipin Kumar

Chapter 5
Analytical Modeling of Parallel Algorithms

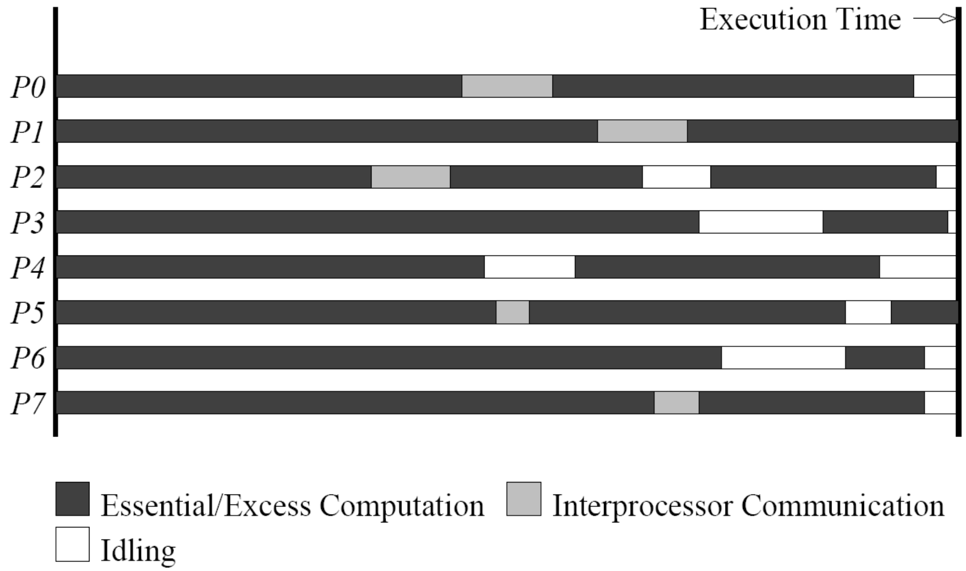
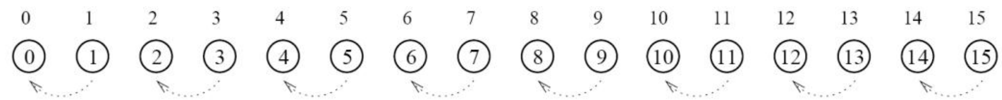
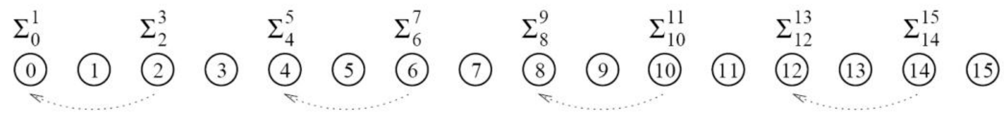


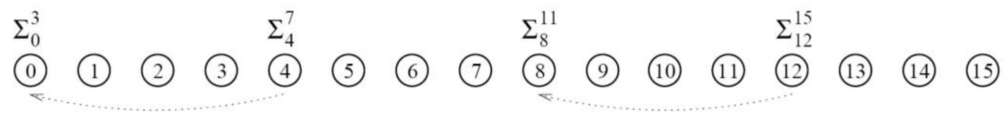
Figure 5.1 The execution profile of a hypothetical parallel program executing on eight processing elements. Profile indicates times spent performing computation (both essential and excess), communication, and idling.



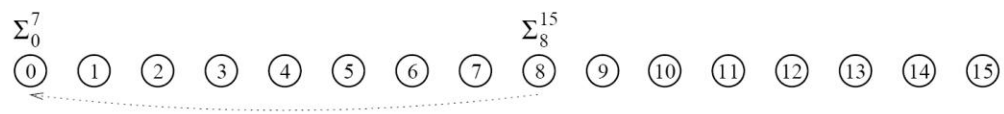
(a) Initial data distribution and the first communication step



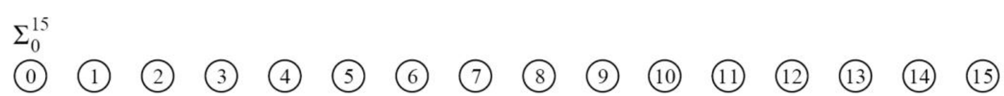
(b) Second communication step



(c) Third communication step



(d) Fourth communication step



(e) Accumulation of the sum at processing element 0 after the final communication

Figure 5.2 Computing the globalsum of 16 partial sums using 16 processing elements. Σ_i^j denotes the sum of numbers with consecutive labels from i to j.

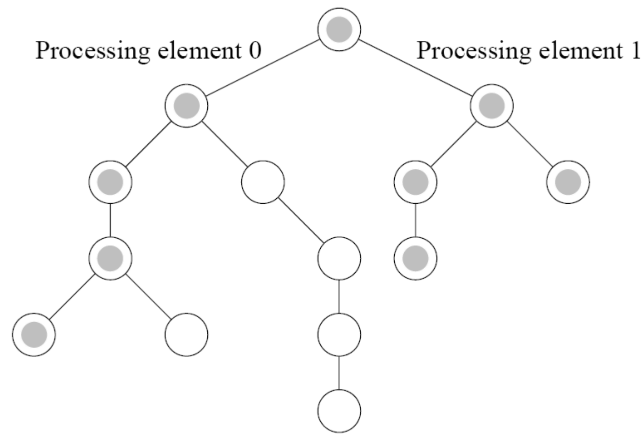


Figure 5.3 Searching an unstructured tree for a node with a given label, 'S', on two processing elements using depth-first traversal. The two-processor version with processor 0 searching the left subtree and processor 1 searching the right subtree expands only the shaded nodes before the solution is found. The corresponding serial formulation expands the entire tree. It is clear that the serial algorithm does more work than the parallel algorithm.

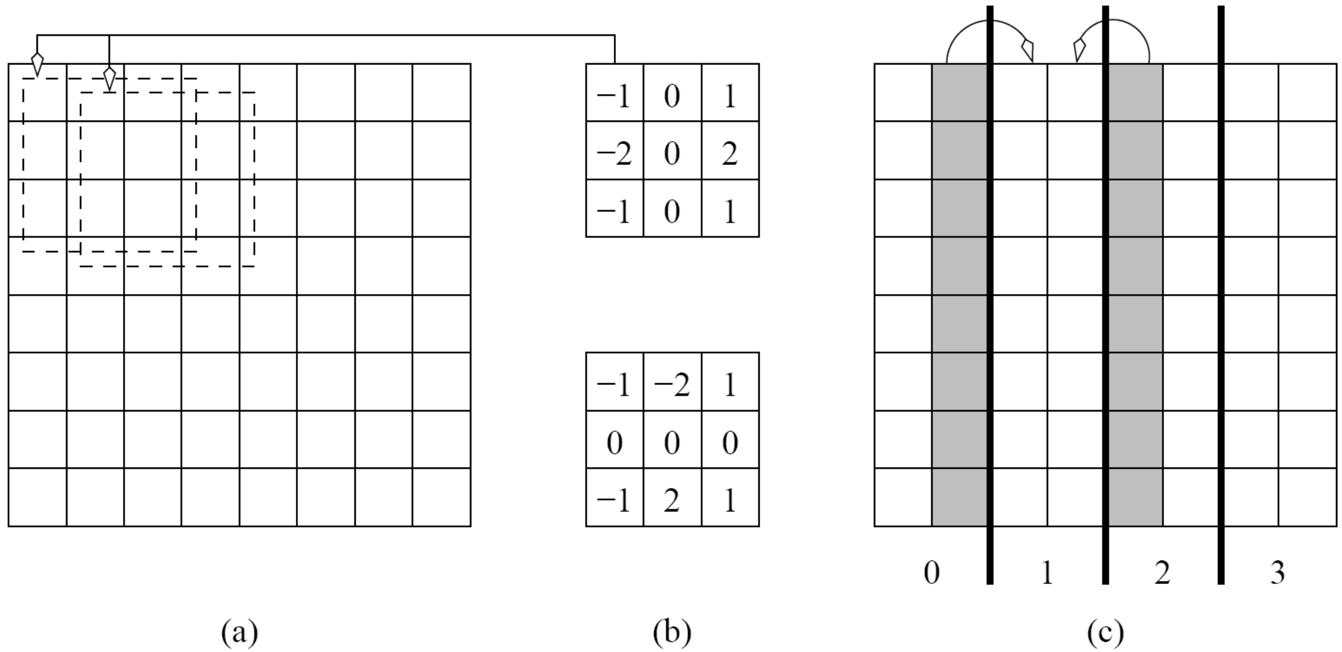
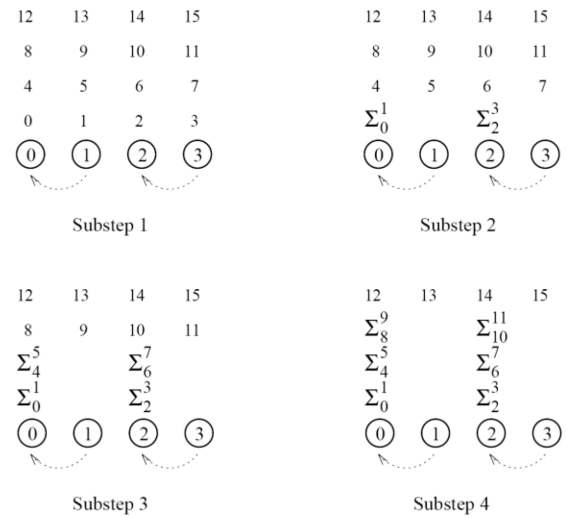
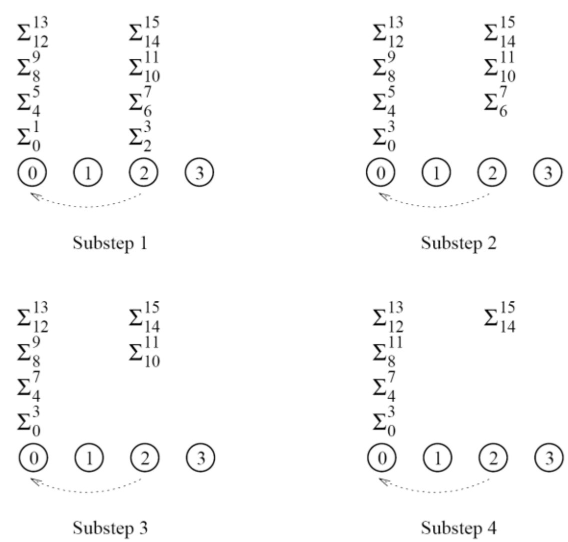


Figure 5.4 Example of edge detection: (a) an 8×8 image; (b) typical templates for detecting edges; and (c) partitioning of the image across four processors with shaded regions indicating image data that must be communicated from neighboring processors to processor 1.

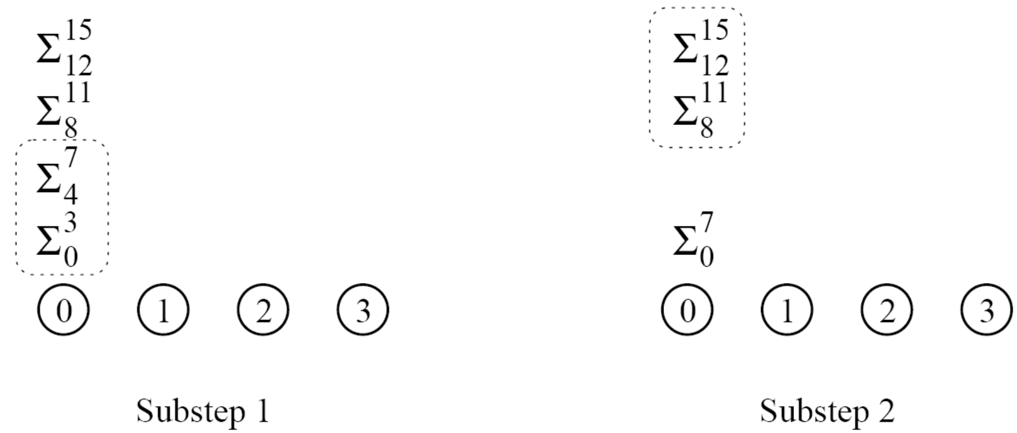


(a) Four processors simulating the first communication step of 16 processors

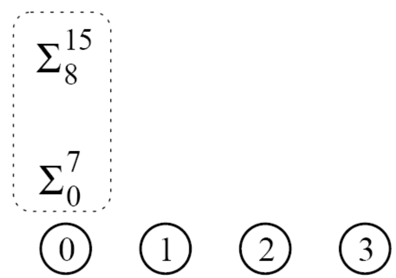


(b) Four processors simulating the second communication step of 16 processors

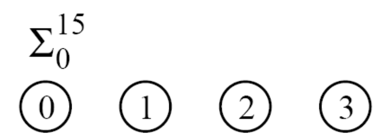
Figure 5.5 Four processing elements simulating 16 processing elements to compute the sum of 16 numbers (first two steps). Σ_i^j denotes the sum of numbers with consecutive labels from i to j.



(c) Simulation of the third step in two substeps



(d) Simulation of the fourth step



(e) Final result

Figure 5.5 (continued) Four processing elements simulating 16 processing elements to compute the sum of 16 numbers (last three steps).

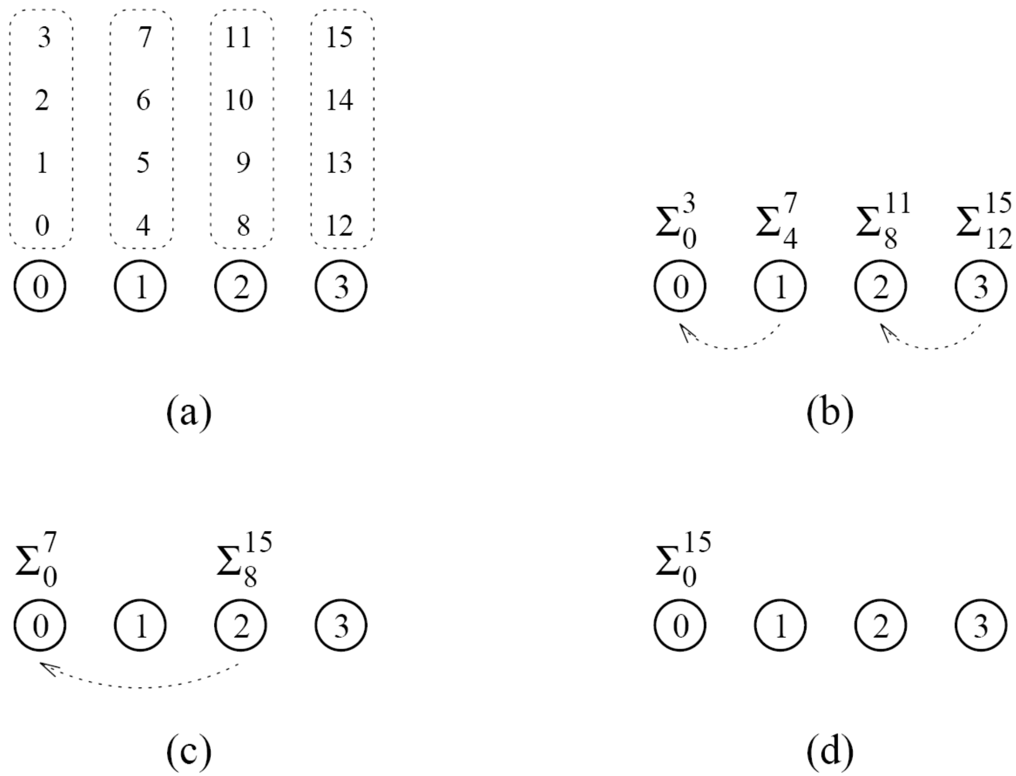


Figure 5.6 A cost-optimal way of computing the sum of 16 numbers using four processing elements.

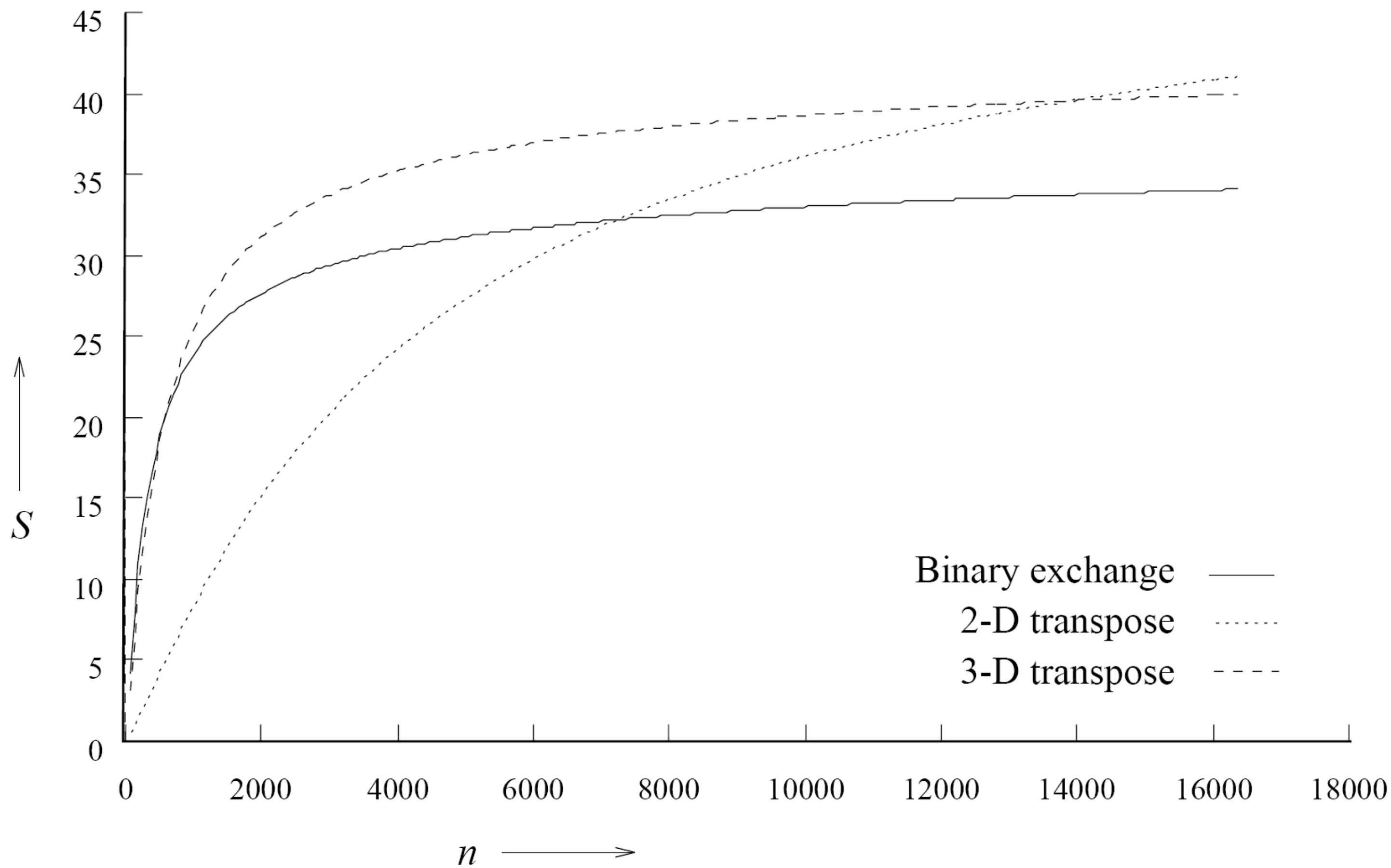


Figure 5.7 A comparison of the speedups obtained by the binary-exchange, 2-D transpose and 3-D transpose algorithms on 64 processing elements with $t_c = 2$, $t_w = 4$, $t_s = 25$, and $t_h = 2$ (see Chapter 13 for details).

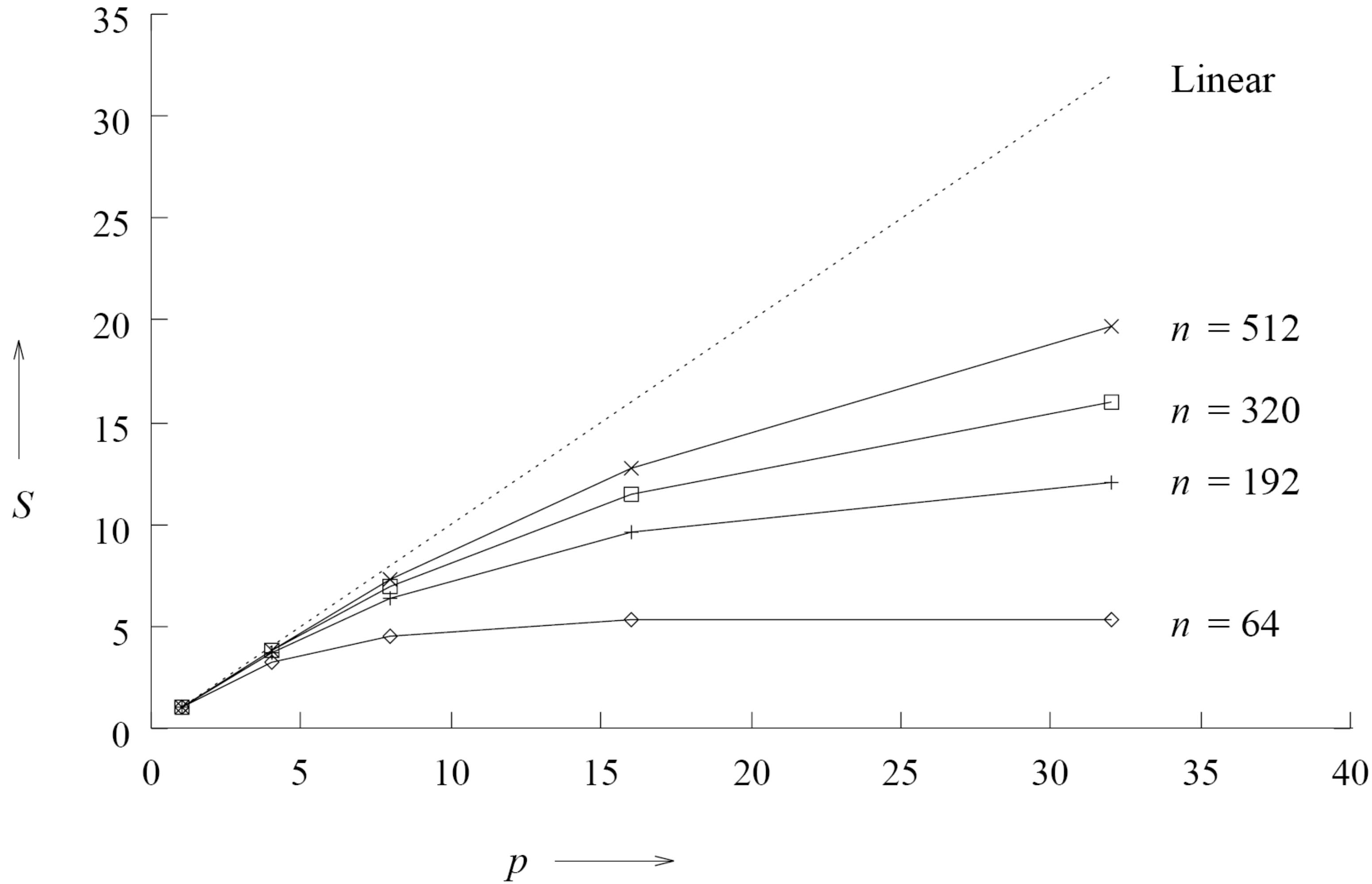


Figure 5.8 Speedup versus the number of processing elements for adding a list of numbers.

Table 5.1 Efficiency as a function of n and p for adding n numbers on p processing elements.

n	$p = 1$	$p = 4$	$p = 8$	$p = 16$	$p = 32$
64	1.0	***0.80***	0.57	0.33	0.17
192	1.0	0.92	***0.80***	0.60	0.38
320	1.0	0.95	0.87	0.71	0.50
512	1.0	0.97	0.91	***0.80***	0.62

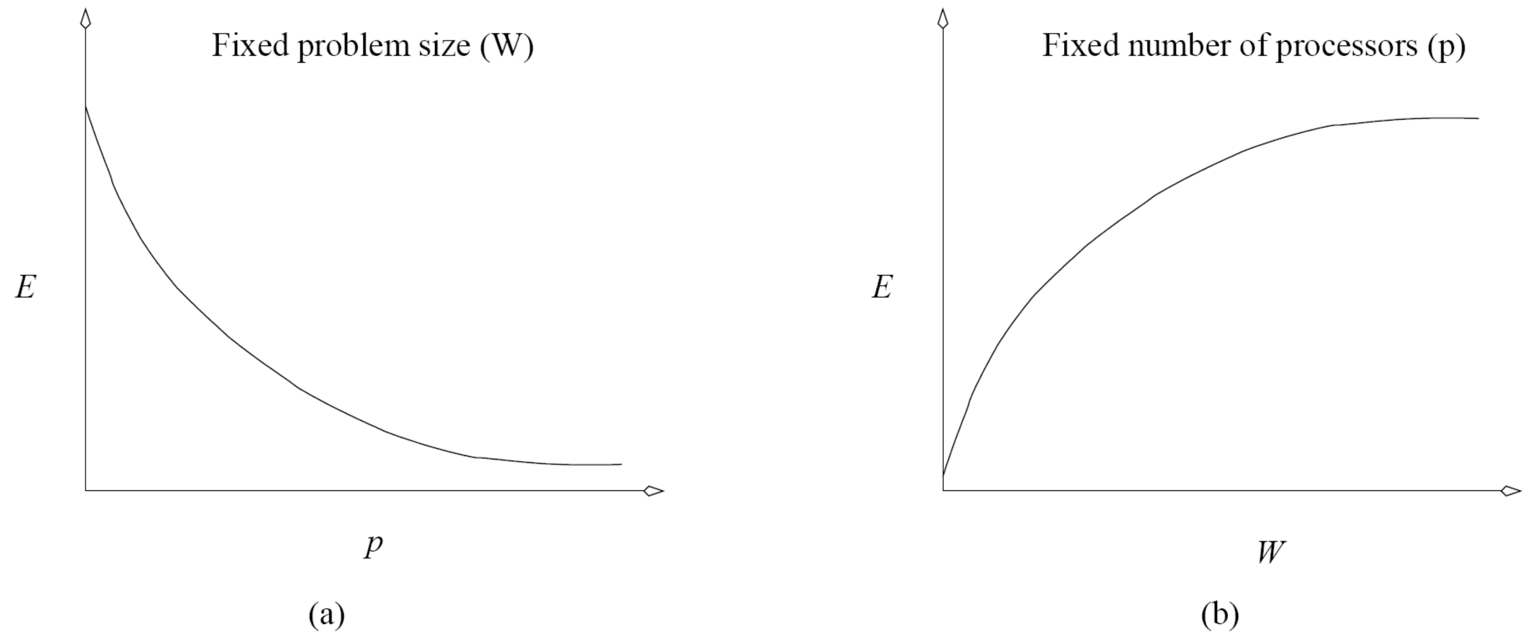


Figure 5.9 Variation of efficiency: (a) as the number of processing elements is increased for a given problem size; and (b) as the problem size is increased for a given number of processing elements. The phenomenon illustrated in graph (b) is not common to all parallel systems.

Table 5.2 Comparison of four different algorithms for sorting a given list of numbers. The table shows number of processing elements, parallel runtime, speedup, efficiency and the pT_P product.

Algorithm	A1	A2	A3	A4
p	n^2	$\log n$	n	$\sqrt{n}$
T_P	1	n	$\sqrt{n}$	$\sqrt{n}\log n$
S	$n\log n$	$\log n$	$\sqrt{n}\log n$	$\sqrt{n}$
E	$\frac{\log n}{n}$	1	$\frac{\log n}{\sqrt{n}}$	1
pT_P	n^2	$n\log n$	$n^{1.5}$	$n\log n$

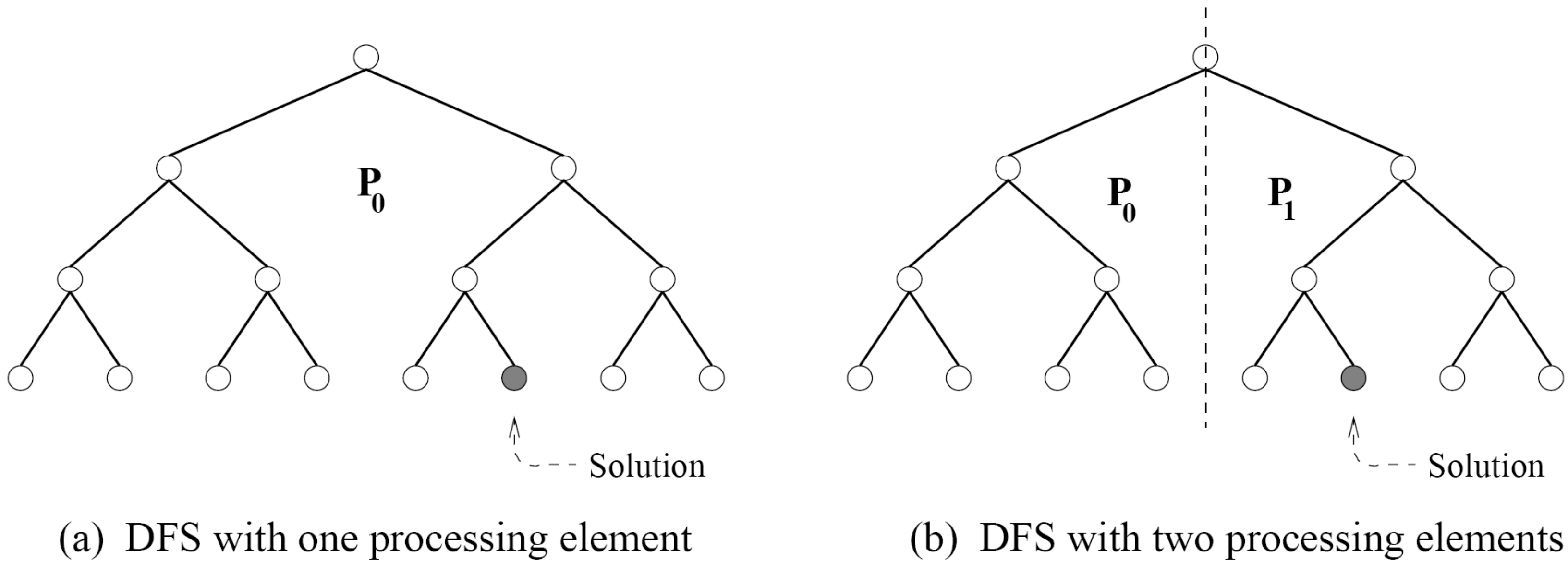


Figure 5.10 Superlinear(?) speedup in parallel depth first search.

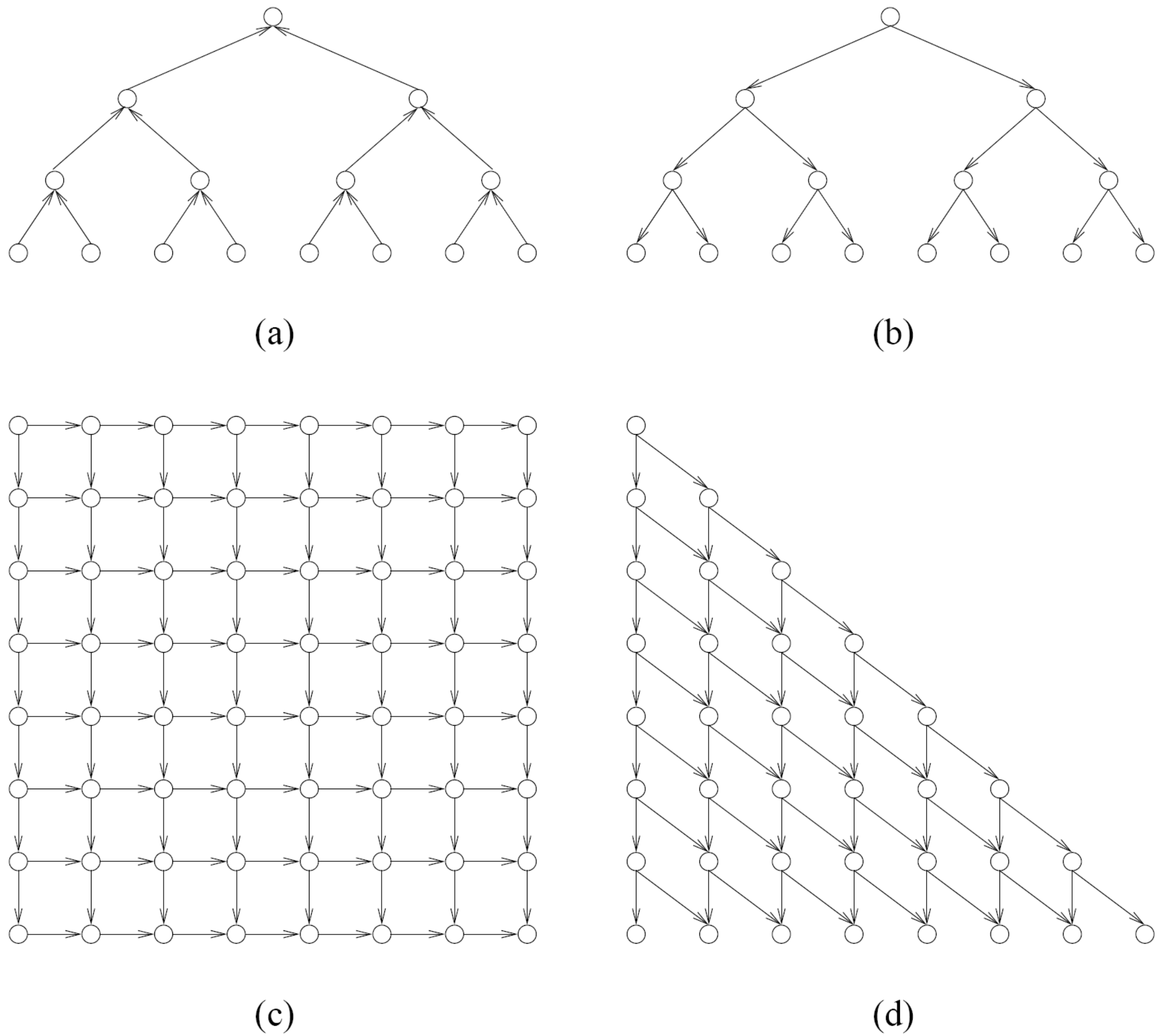


Figure 5.11 Dependency graphs for Problem 5.3.