Birla Institute of Technology and Science

CS - F422

Parallel Computing
Assignment-2

Anirudh Kumar Bansal
2014A7PS081P

Lakshit Bhutani
2014A7PS095P

18 April 2017

Contents

1 Problem Statement

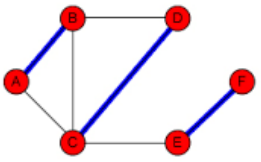
- Design a hybrid Branch-and-Bound Skeleton scheme for a target platform that is a cluster of workstations.
- Choose two hard / pseduo-hard problems (e.g. 0/1 KnapSack, TSP, Graph Coloring, Factorization) and encode each of them using the Branch and Bound skeleton .
- Implement your solution using MPI for message passing programming on the cluster and OpenMP for shared memory programming within a single node. Implement your solution in two steps:
  1. MPI version where each process is single-threaded
  2. OpenMP version to run in a multicore node

1.1 Specific Problems

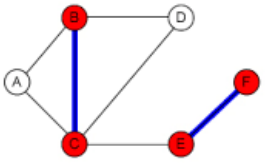
The two specific problems chosen for the assignment are Minimum Vertex Cover problem and Minimum Assignment problem.

1.1.1 Minimum Vertex Cover

Consider a graph $G = (V, E)$. $S \subseteq V$ is a vertex cover of G iff $\forall (u, v) \in E : u \in S \vee v \in S$. Minimum vertex cover is the problem of finding a set S with minimum $|S|$.



(a) Vertex cover of size 6



(b) Vertex cover of size 4

Figure 1: Vertex Cover of varying sizes. One of the minimum vertex cover is $S = \{A, D, E\}$ with $|S| = 3$

1.1.2 Minimum Assignment Problem

The problem instance has a number of agents and a number of tasks. Any agent can be assigned to perform any task, incurring some cost that may vary depending on the agent-task assignment. It is required to perform all tasks by assigning exactly one agent to each task and exactly one task to each agent in such a way that the total cost of the assignment is minimized.

	Job 1	Job 2	Job 3	Job 4
A	9	2	7	8
B	6	4	3	7
C	5	8	1	8
D	7	6	9	4

Figure 2: Minimum assignment with cost $2 + 6 + 1 + 4 = 13$

2 Vertex Cover problem

At each node denoting a partial solution a list of vertices to be included in the vertex cover and the list of edges considered so far is stored. The branch and bound template for vertex cover involves the following two steps :

2.1 Branch

A new edge which has not been covered in the partial solution is added to the solution edge set. If the edge is already covered by the vertices in the partial solution then the vertex set remains unchanged else one of the end points of the edge is added to the vertex set.

2.2 Bound

The lower bound at any given node denoting a partial solution (denoting a lower bound on the minimum possible number of vertices in any vertex cover that could be obtained by extending the partial solution) is obtained by adding up the current number of vertices with the lower bound on extra number of vertices required to cover the uncovered edges.

Lemma 2.1. *If there are m uncovered edges and n remaining vertices $v_1, v_2, v_3...v_n$, then a lower bound on the number of vertices required to cover these edges is $\frac{m}{\max_{i=1}^{n} deg(v_i)}$ where $deg(v_i)$ denotes the degree of vertex i.*

Proof. : Each vertex v_i covers $deg(v_i)$ edges independently. However it is possible for an edge to be covered by more than one vertex. Thus the total number of edges covered by the vertices $v_1, v_2, v_3...v_n$ is at most equal to the sum of their degrees i.e.

$$m \leq \sum_{i=1}^{n} deg(v_i) \leq n \cdot \max_{i=1}^{n} deg(v_i)$$

or

$$n \geq \frac{m}{\max_{i=1}^{n} deg(v_i)}$$

□

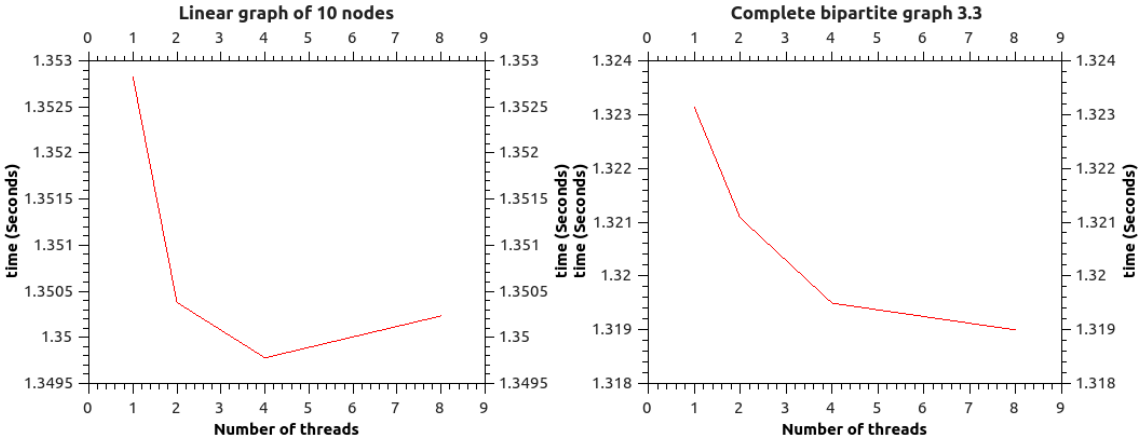
2.3 Results

2.3.1 Shared Memory Design

The program to find vertex cover was run on system with four cores and varying number of threads for various types of graphs.

Type of graph	Number of threads	Time taken (seconds)
Linear graph of 10 nodes	1	1.352827
	2	1.350379
	4	1.349778
	8	1.36
$K_{3,3}$	1	1.323145
	2	1.32108
	4	1.319494
	8	1.318994

Table 1: Performance on shared memory architecture



(a) Linear graph of 10 nodes (b) $K_{3,3}$

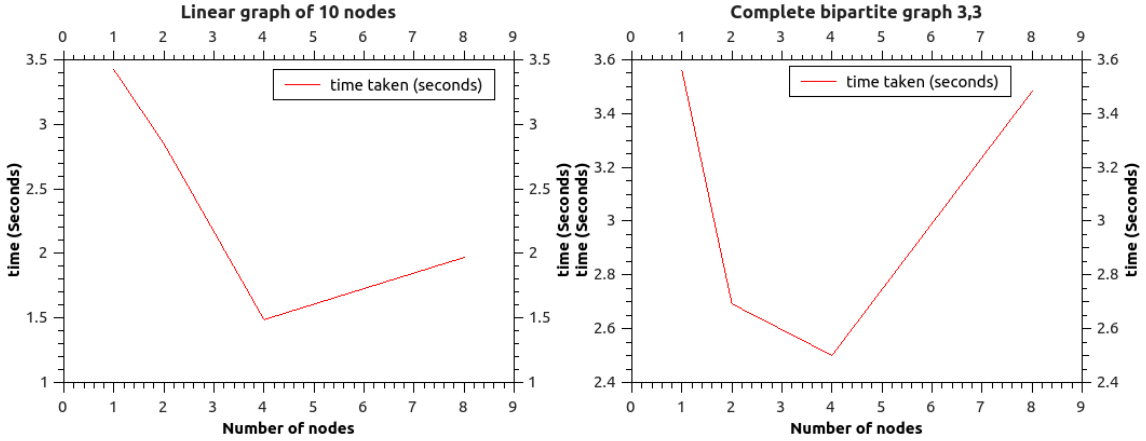
Figure 3: Plot of performance on shared memory architecture

2.3.2 Message Passing Design

The program to find vertex cover was run on four core processor with multiple number of independent processes for various types of graphs.

Type of graph	Number of processes	Time taken (seconds)
Linear graph of 10 nodes	1	3.422137
	2	2.849857
	4	1.486590
	8	1.968621
$K_{3,3}$	1	3.561021
	2	2.692162
	4	2.500457
	8	3.482213

Table 2: Performance on message passing architecture



(a) Linear graph of 10 nodes (b) $K_{3,3}$

Figure 4: Plot of performance on message passing architecture

3 Assignment problem

At each node denoting a partial solution the total number of agents assigned a task and the corresponding list of tasks is stored . The branch and bound template for assignment problem involves the following two steps :

3.1 Branch

A new agent which has not been assigned a task yet in the partial solution is added to the agent set. The new agent is assigned one of the tasks what are available amongst all such tasks.

3.2 Bound

The lower bound at any given node denoting a partial solution (denoting a lower bound on the least possible total cost in any task assignment that could be obtained by extending the partial solution) is obtained by adding up the current cost with

the lower bound on the additional cost obtained in assigning tasks to the remaining agents.

Lemma 3.1. *Let cost[i][j] denote the cost associated with assigning task j to agent i. If there are $n-k$ remaining agents $k+1, k+2, ...n$ and $n-k$ remaining tasks $j_{k+1}, j_{k+2}, ...j_n$, then a lower bound on the total cost incurred in the assignment is*

$$\sum_{i=k+1}^{n} minc_i$$

where

$$minc_i = \min_{a=k+1}^{n} cost[a][i]$$

.

Proof. : Consider any assignment of the remaining tasks to the remaining agents , then since each task j_k is assigned to any one agent, the cost associated with this task-agent assignment is at least $minc_k$. Thus as the total additional cost is the sum of all such task-agent assignment costs (for each task) , a lower bound on the total additional cost C is given by the sum of all such *minc* values i.e

$$C \geq \sum_{i=k+1}^{n} minc_i$$

. □

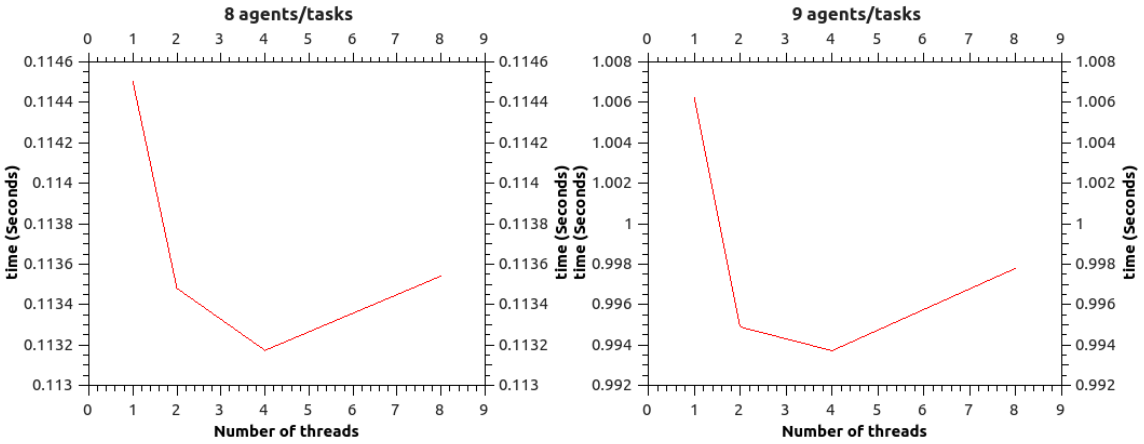
3.3 Results

3.3.1 Shared Memory Design

The program to find minimum cost assignment was run on system with four cores and varying number of threads for different number of agents / tasks.

Number of agents / tasks	Number of threads	Time taken (seconds)
8	1	0.114503
	2	0.113480
	4	0.113173
	8	0.113541
9	1	1.006228
	2	0.994889
	4	0.993714
	8	0.997796

Table 3: Performance on shared memory architecture



(a) 8 agents/tasks (b) 9 agents/tasks

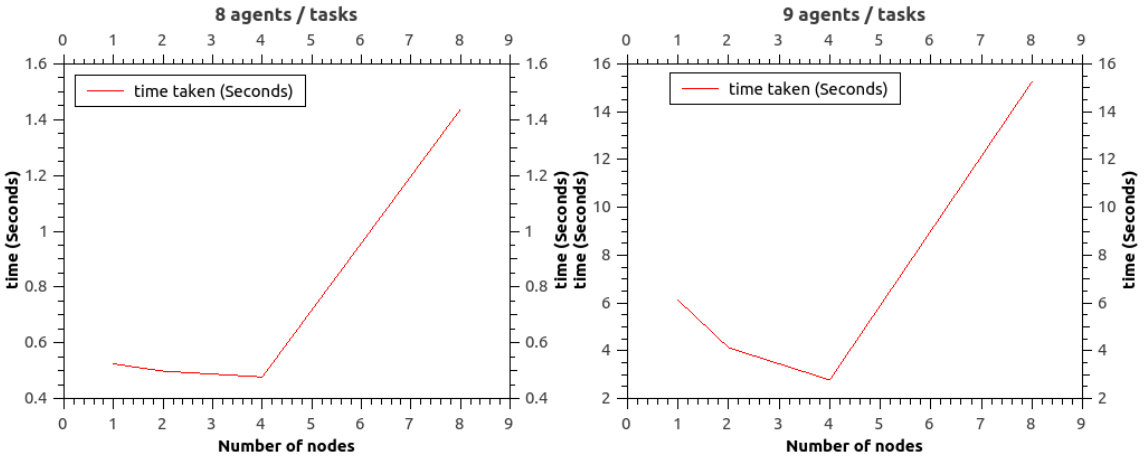
Figure 5: Plot of performance on shared memory architecture

3.3.2 Message Passing Design

The program to find minimum cost assignment was run on four core processor with multiple number of independent processes for varying number of agents / tasks.

Number of agents / tasks	Number of processes	Time taken (seconds)
8	1	0.523980
	2	0.497898
	4	0.476905
	8	1.437256
9	1	6.113277
	2	4.123094
	4	2.773906
	8	15.252611

Table 4: Performance on message passing architecture



(a) 8 agents / tasks (b) 9 agents / tasks

Figure 6: Plot of performance on message passing architecture

4 Deliverables

The folder named *assignment2.zip* has the structure :

- omp
  - vertex_cover
  - assign
- mpi
  - vertex_cover
  - assign
- report.pdf

Each leaf level folder contains *code*, *sample inputs*, *readme* and *makefile*. Please refer to *readme* for exact usage.