Solving the Travelling Salesman Problem on Shared and Distributed Memory Multiprocessor Systems

A. L. Ignatyev*, M. A. Posypkin†, I. Kh. Sigal‡

*Computing Center RAS, ignatyev.alexander@gmail.com
†Institute for System Analysis RAS, mposypkin@mail.ru
†Computing Center RAS, sigal@ccas.ru

Parallel computers can roughly be divided into two families: systems with shared memory and systems with distributed memory organization. Shared memory architecture uses a single address space. Systems based on this concept allow processor communication through variables stored in a common address space. Another major architecture for parallel computers employs a scheme by which each processor has its own memory module. Such a distributed-memory multiprocessor is constructed by connecting each component with a high-speed communications network. Processors communicate to each other over the network.

In this paper we consider the famous *Traveling Salesman Problem* (TSP) [1], formulated as follows: given a set of cities along with the cost of travel between each pair of them, the problem is to find the cheapest way of visiting all the cities and returning to the starting point. Due to the high computational complexity the TSP is a frequent target for parallel and distributed computing.

We selected the Branch-and-Bound approach to solve the TSP problem. The Branch-and-Bound (B&B) is a general name for methods to split an initial problem into subproblems which are sooner or later eliminated by bounding rules. Bounding rules determine whether a subproblem can yield a solution better than the best solution found so far. The latter is called incumbent solution. Bounding is often done by comparing lower and upper bounds: a subproblem can be pruned if the lower bound for its objective is larger or equal to the current upper bound, i.e. incumbent solution. Thus the quality upper bound significantly reduces the search space and in some cases leads to dramatic performance improvements. Fortunately Branch-and-Bound is highly suitable for parallel and distributed computing: after splitting the parts of the solution space can be processed simultaneously.

The B&B scheme for TSP was implemented within the BNB-Solver [2] framework. The BNB-Solver is a library for solving discrete and global optimization problems on serial machines, shared memory and distributed memory parallel computers. BNB-Solver follows the standard approach taken in software frameworks for discrete and global optimization: the problem-specific and problem-independent (core) parts are separated and implemented independently. Coupling of problem-specific and problem-independent parts is done via C++ template mechanism. Problem-specific branching operations, bounding rules and type definitions for subproblems and feasible points are encapsulated by a problem factory class. This class is substituted as a template argument within specializations of core classes.

Table 1: Time (sec) for different problem instances and different lower bound procedures obtained on 64 cores.

Problem	ftv33	ftv38	ftv47	ftv55
Matrix reduction	0.12	0.21	3.61	108.02
Assignment problem	0.11	0.08	1.64	5.03

Experiments were run on a MVS100K [3] computational cluster consisting of 990 SMP nodes with two 4-core Intel Xeon processors in each. The first series of experiments was aimed at choosing best lower bound procedure. We implemented two procedures for calculating lower bound. The first one is based on *matrix reduction*. The second uses an *assignement problem*. The bound based on a assignment problem is tighter but requires more computational efforts. We compared both approaches on several assymetric TSP instances from TSPLIB [4]. The results presented in Table 1 suggest that for asymmetric TSP the lower

Table 2: Time (sec) of distributed memory implementation of ftv90 instance

1	2	4	8	16	32	64
636.8	96.77	35.25	41.33	26.63	19.36	8.17

Table 3: Time (sec) of shared memory implementation for ftv47 instance

1	2	3	4
7.07	3.98	2.67	2.23

bound based on assignement problem is favorable. We selected this approach for further experiments. Table 2 demonstrates running time for 90-cities problem from the collection [4]. The number of processors varies from 1 to 64. The obtained results confirm good scalability for distributed memory implementation: the running time decreases from 636.8 to 8.17 seconds. The table also demonstrates the well known phenomena — so called *search anomalies*: the ratio of number of processors and running time is not constant. It is explained by different number of expanded subproblems that depends on an order of search tree traversal.

Running time of shared memory implementation for ftv47 problem obtained on Intel Core 2 Quad Q9650 3.0 GHz is presented in the Table 3. Number of threads varies from 1 to 4. Distributed memory implementation scales better w.r.t. shared memory one. This fact indicates problems with threads contention for shared resources.

Table 4 lists running time obtained with distributed memory implementation on 64 computational cores. Eight instances from [4] were taken with the number of cities ranging from 44 to 443.

In future we plan to improve the shared memory implementation by applying more intellectual memory management scheme and implementing tree-based storage for subproblems. BNB-Solver library supports hybrid approaches combining exact (B&B) and heuristic methods. Hybrid approaches showed very good performance for knapsack problem [5]. We are going to try hybrid approaches for TSP problem as well.

Table 4: Running time (sec) for different TSP instances

Ftv 44	Ftv90	Ftv100	Ftv110
0.09	4.1	104	888
rbg323	rbg358	rbg403	rbg443
277	553	1259	2523

Acknowledgements. This work was supported by RFBR grant 08-07-00072-a.

References

[1] E. L. Lawler , J. K. Lenstra, A. H. G. Rinnooy Kan, D. B. Shmoys, eds. 1985. The travelling salesman problem. John Willey & Sons, Chichester, UK.

[2] Y. Evtushenko, M. Posypkin, I. Sigal, A framework for parallel large-scale global optimization // Computer Science - Research and Development (in print)

[3] Joint Supercomputer Center. http://www.jscc.ru

[4] G. Reinelt, TSPLIB - A Traveling Salesman Problem Library // ORSA Journal on Computing. 1991. N. 3. P. 376-384.

[5] M. Posypkin, I. Sigal, A combined parallel algorithm for solving the knapsack problem, Journal of Computer and Systems Sciences International, Vol. 47, N. 4, p.543–551.