OpenMP Parallelization Strategy for Bee Colony Optimization

T. Jaksić[1], T. Davidović[1], and P. Jain[2]

[1] Mathematical Institute of the Serbian Academy of Sciences and Arts, Belgrade, Serbia
{tatjana.tanjad}@mi.sanu.ac.rs
[2] Birla Institute of Technology and Science, Pilani, Rajasthan, India
jainparvesh.321@gmail.com

1 Introduction

The Bee Colony Optimization (BCO) algorithm is a meta–heuristic that belongs to the class of biologically inspired stochastic swarm optimization methods, based on the foraging habits of bees in nature. It has successfully been applied to various combinatorial optimization problems [2]. The basic idea behind BCO is to create a multi-agent system (colony of artificial bees) capable of successfully solving difficult combinatorial optimization problems. In its basic form, BCO belongs to the class of constructive methods: it builds solutions from scratch through a sequence of execution steps, unlike the local search-based meta-heuristics, which perform iterative improvements of the current best solution. BCO operates on a population of solutions, and therefore we expect it to represent a good basis for parallelization. The main contribution of this work is the development of new and efficient parallelization strategy for BCO. We propose parallelization strategy for a shared memory multiprocessor architecture under the Open Multi Processing (OpenMP) communication protocol. Our strategy is based on fine-grained parallelization techniques. We obtained super linear speedup while preserving the solution quality.

2 Parallelization technique

Basic ideas regarding parallel execution of BCO were reported in [1] using high-level parallelization implemented with MPI communication protocol (together with review of the MPI parallelization of other swarm-inspired algorithms). To the best of authors' knowledge, parallel version for shared memory architectures is offered only for ABC algorithm ([3]), however, not using fine-grained techniques.

Portions of work under for-loops are parallelized in such a way that the total workload is being equally distributed between individual processors. The main difficulty that we faced when treating fine-grained parallelization techniques on shared–memory multiprocessor systems was the following: the portion of work to be performed in parallel is too small, thus the extensive use of CPU time for creating threads and their synchronization exceeds the benefits of parallel execution. To overcome this problem, we create threads at the beginning of the algorithm execution and keep them "alive" till the end. We use barrier directives, critical regions and sequentially executed parts to synchronize individual processor calculations and ensure correct execution of dependent computations in the code. We refer to this approach as OBCO.

3 Experimental evaluation

The experimental evaluation is performed on the problem of scheduling independent tasks to identical machines. Our starting point is the sequential implementation of the BCO algorithm proposed in [2]. The proposed parallelization strategy results in a significant speedup, as well as an improvement in the quality of the final solution as it is reported in Table 1.

In the first column of Table 1, the number of parallel threads q, executing OBCO, is given. The second and third column contain minimum and maximum running (wall clock) time of OBCO over q threads. Forth column contains average running time out of 100 repetitions. Fifth column contains minimum value of the objective function (schedule length). The corresponding maximum

Table 1. OBCO Scheduling results – test problem Iogra100_12 with known optimal solutions

q	min. time	max. time	av. time	min. OBCO	max. OBCO	av. OBCO	min. min_t
1	177.2	188.55	185.58	808	813	810.3	9.11
2	49.23	78.32	63.42	808	812	809.9	7.01
3	26.57	55.27	40.26	806	812	809.9	3.72
4	19.71	35.85	27.92	807	812	810.1	2.24
5	18.83	34.67	27.66	806	813	810.3	2.51
6	16.04	29.12	24.24	806	812	810.3	1.39
7	13.45	26.02	21.34	808	812	810.4	1.83
8	11.28	21.56	18.25	808	814	810.7	1.19

value is presented in the sixth column of this table. The average schedule length represents the content of column seven. The average time required to obtain best solution by OBCO is given in the last column. From the results presented in Table 1 we conclude that, for the Iogra100_12 example, the quality of the solution is either improved or preserved with respect to the sequential execution of BCO. At the same time the total time is reduced, and we obtained complete scalability, as shown in Figure 1.

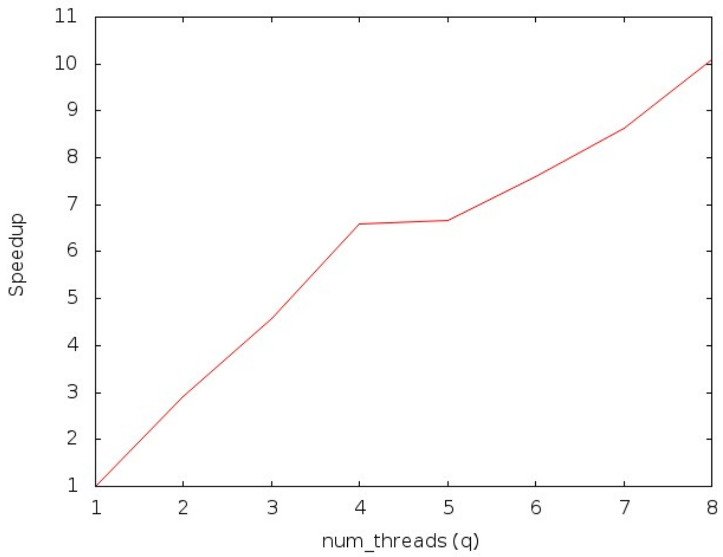

Fig. 1. Speedup of OBCO code

Gained speedup is limited by the number of processors in the available shared memory system. Our future work will include combination of high-level and low-level parallelization on distributed systems that allow implementation of MPI-OpenMP hybrid model.

References

1. Davidović, T., Jakšic, T., Ramljak, D., Šelmić, M., Teodorović, D.: MPI Parallelization Strategies for Bee Colony Optimization, *Optimization*, 2012 (submitted).
2. Davidović, T., Šelmić, M., Teodorović, D., Ramljak, D.: Bee colony optimization for scheduling independent tasks to identical processors, *Journal of Heuristics*, 2012 (in press) DOI:10.1007/s10732-012-9197-3.
3. Narasimhan, H.: Parallel Artificial Bee Colony (PABC) Algorithm, *Proc. World Congress on Nature and Biologically Inspired Computing* (NaBIC '09), Coimbatore, 2009, 306-311.