3rd International Conference on Industrial Engineering and Industrial Management
XIII Congreso de Ingeniería de Organización
Barcelona-Terrassa, September 2nd-4th 2009

A prototype of parallel hybrid algorithm

Jose Miguel León Blanco[1], Jose Manuel Framiñan[1], Pedro Luis González Rodríguez[1], Jose Luis Andrade Pineda[1], Rafael Ruiz Usano[1]

[1] Dpto. de Organización de Empresas. Escuela Técnica Superior de Ingenieros de Sevilla. Universidad de Sevilla. Cº. Descubrimientos s/n, 410952. Sevilla. miguel,@esi.us.es, jose@esi.us.es, pedroluis@esi.us.es, jlandrade@esi.us.es, usano@esi.us.es

Keywords: Branch and Bound, CLM, Parallel algorithms, metaheuristics,

1. Introduction

Parallel algorithms have focused an increase interest due to advantages in computation time and quality of solutions when applied to industrial engineering problems. In this communication*, we present a prototype for a parallel hybrid algorithm combining an exact method like Branch and Bound (B&B) and a metaheuristic like Complete Local Search (CLM), Ghosh and Sierksma (2002), based in the sequential algorithm Bound Driven Search (BDS), Framinan and Pastor (2008).

2. Parallel hybrid algorithms

It has been demonstrated that cooperation between different types of algorithms leads to better quality solutions and more robust methods, Talbi (2002), Basseur et al. (2004). Although there are many sequential algorithms combining metaheuristics or metaheuristics and exact methods, there are few implementations of parallel hybrid algorithms combining exact and approximate methods. In the survey carried out by Jourdan et al. (2008) there are 61 references of hybrid methods but only 3 describe parallel hybrid methods combining exact and approximate algorithms. The only exact method mentioned is branch and bound.

Another survey is the one by Puchinger & Raidl (2005). This one refers an architecture of hybrid parallel algorithms based on multiagent systems. Each agent is an algorithm, so the system can be executed in one or more processors of a computer, in a cluster of workstations or in a distributed set of computers. The architecture is known as teams of agents in two similar approaches, Asynchronous Teams (A-Teams) as described by Talukdar et al. (1998), by Talukdar (1998) and by Talukdar (1992) or TEams for Cooperative Heterogeneous Search (TECHS) as presented by Denzinger & Offerman (1999).

3. Parallel hybrid prototype

In this prototype (Figure 1), we extend BDS algorithm employing several parallel processes. Half of them are dedicated to B&B and the rest employ CLM. The first B&B process generates an initial solution using a constructive heuristic (step 1 in figure 1). This process explores the tree of solutions (step 2) and generates three lists of solutions: complete solutions (LIVE), partial solutions (P_LIVE) and solutions belonging to pruned branches

* This work stems from the participation of the authors in a research project funded by the Spanish Ministry of Science, grant DPI2007-61345, title Advanced Systems for Integrated Order Management, and SCOPE, grant P08-TEP-3630, funded by the Andalusian Government.

(FORBIDDEN). This first exploration of the tree must generate a number of partial (P_LIVE) solutions greater than the number of B&B processes and a number of complete (LIVE) solutions greater than the number of CLM processes. This strategy guarantees that each of the processes will be able to start its search from a different solution.

P_LIVE list is broadcasted to the rest of B&B processes together with FORBIDDEN and LIVE list (step 3). The lists broadcasted to CLM processes are LIVE and FORBIDDEN (steps 4 and 5).

From now on (steps 6 and 7), each B&B process iterates exploring branches of the tree (steps 10 and 11) in a deep-first strategy. At the beginning of each iteration, the process deletes as many partial solutions from P_LIVE as the number of the process. This strategy prevents different B&B processes to explore the same branch of the tree.

At the same time (steps 8 and 9) CLM processes iterate exploring the neighbourhood (steps 12 and 13) of the solutions of LIVE list. At the beginning of each iteration, the process deletes as many solutions from LIVE as the number of the process. This strategy prevents different CLM processes to begin exploring the same neigbourhood.

We employ the concept of energy conservation as described in Pacheco (1997) for a tree search algorithm. When one process ends its work, it looks for more work looking for pending messages from other process. If there's a message, it receives data and continues working. This strategy is necessary to obtain an efficient parallel algorithm, avoiding both waiting processes and overloaded processes.

When a process ends its exploring, it looks for messages pending to be received from other processes. If there is a message, the process will receive it and store the received solutions if they are new or improve the best current solution. The process will continue receiving messages till there are no more. After receiving messages, the process will send the new solutions found in the exploration.

We extend here this concept taking into account that there are processes exploring a tree and others exploring a neighbourhood. The kind of information required is different.

When a B&B process ends an iteration finding a complete solution, it sends the new found solutions to the rest of processes as they need them: B&B processes need to know if the upper bound has been improved by another process and they also need to know if there are new pruned branches. CLM processes need to know new complete solutions and solutions that cannot be explored. Thereby, B&B processes send new solutions added to P_LIVE, LIVE and FORBIDDEN lists to other B&B processes (steps 14 and 15). They send new solutions added to LIVE and FORBIDDEN lists to CLM processes (step 16).

CLM processes end an iteration due to three conditions: they have explored all the neighbourhood, they have run out of memory or they have explored many solutions without any improvement in the objective function. At each iteration, CLM processes send to B&B processes the best found solution if it is better than previous sent solution to help them improve the upper bound. CLM processes send to other CLM processes the best solution and the solutions added to DEAD list in the last iteration, to prevent them to explore the same neigborhood. Thereby, CLM processes send new solutions added to DEAD list to other CLM processes (steps 17 and 18). They send the best solution to B&B processes (step 19).

This is the same strategy used by Denzinger & Offerman (1999). B&B processes send to the rest, positive and negative information. Positive information is included in P_LIVE and LIVE lists and negative information, pruned branches of the tree, is included in FORBIDDEN list.

In the other hand, CLM processes send positive information, the best found solution and negative information in DEAD list, all explored solutions.

When a process receives a message containing solutions found by other processes, it must update its knowledge of the search (steps 20-23). B&B processes should update the upper bound with the solutions received from CLM and the lists from B&B processes and consequently the content of P_LIVE and FORBIDDEN lists. Also, CLM processes should update LIVE list deleting the solutions similar to those received in FORBIDDEN list and DEAD list with the solutions received from other CLMs.

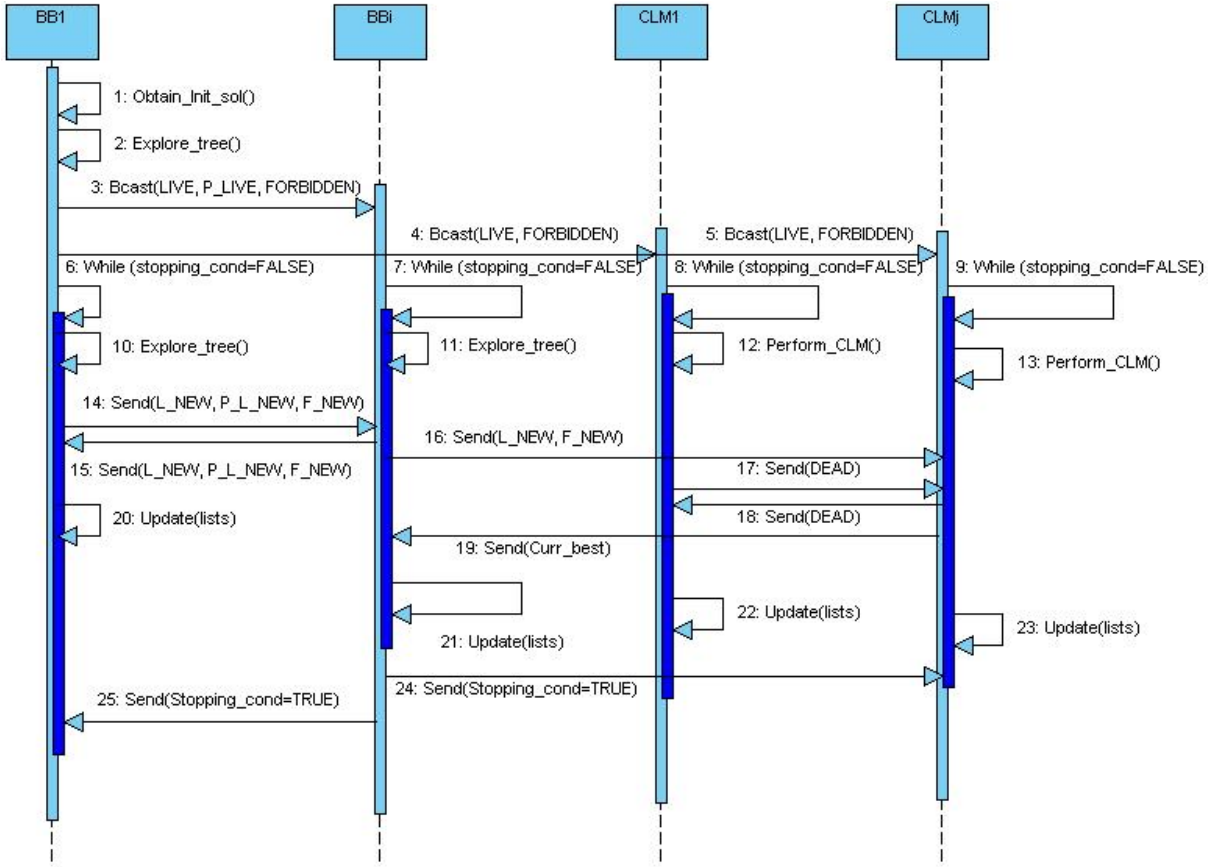

Figure 1. Sequence diagram of parallel BDS algorithm

To finish the algorithm, there are two possible conditions. The obvious one is that one of B&B processes has found the exact solution. The second one is that a prefixed amount of time has passed since the beginning of the algorithm.

The first condition is verified when all B&B processes has emptied their partial solutions lists. To verify this condition, each B&B process could use an binary array composed of the signal of empty list send by other processes. When a process receives a message telling another one has emptied P_LIVE list, insert this condition into the array and checks if all B&B processes have sent this condition. In that case, it will send a stopping message to all the processes (steps 24 and 25). In the other hand, when a process receives new partial solutions, it must update this situation in the array.

The second condition is verified by each process simply using the system clock to check if time elapsed from the beginning of the algorithm has surpassed the maximum time available for the exploration.

4. Future work

The next step in our research is the implementation of this algorithm in a cluster of workstations employing MPI as the communication mechanism. Our challenge will be to develop an efficient algorithm applied to the permutation flow shop problem.

The main difficulties in this implementation will be to obtain an efficient algorithm due to the communication policy used. The first solutions sent by the first B&B process will use MPI_Bcast() function, and can be easily implemented. The rest of messages must be sent in a non-blocking way so the process can continue exploring the tree or the neigbourhood and sending messages to other processes simultaneously.

In our first experiences, we have found it difficult to make processes notice that there are pending messages, at least in small problems, when computation ends quickly and there is no time to read all the communication. Another issue happens when a process has found many solutions and the receiving process cannot read all the message before the communication is interrupted by another communication.

References

Basseur, M.; Lemesre, J.; Dhaenens, C.; Talbi, E.G. (2004). Cooperation between branch and bound and evolutionary approaches to solve a bi-objective flow shop problem. Springer Berlin / Heidelberg. Experimental and Efficient Algorithms. Third International Workshop. . Angra dos Reis, Brazil. Springer Berlin / Heidelberg. pp. 72-86.

Denzinger, J.; Offerman, T. (1999). On cooperation between evolutionary algorithms and other search paradigms. Proceedings of Congress on Evolutionary Computation CEC1999. Washington, DC, USA: IEEE, 1999. 2317-2324.

Framinan, J.M.; Pastor, R. (2008). A proposal for a hybrid meta-strategy for combinatorial optimization problems. Journal of Heuristics, Vol. 14, No. 4, pp. 375-390.

Ghosh, D.; Sierksma, (2002). Complete Local Search with Memory. Journal of Heuristics, Vol. 8, No. 6, 571-584.

Jourdan, L.; Basseur, M.; Talbi, E.G. (2008). Hybridizing exact methods and metaheuristics: A taxonomy. European Journal of Operational Research In Press, Corrected Proof. doi:10.1016/j.ejor.2007.07.035.

Pacheco, P.S. (1997). Parallel Programming with MPI. Morgan Kaufmann Publishers, Inc. / Elsevier.

Puchinger, J.; Raidl, G.R. (2005). Combining metaheuristics and exact algorithms in combinatorial optimization: a survey and classification. IWINAC, pp. 41–53.

Talbi, E.G. (2002). A taxonomy of Hybrid Metaheuristics. Journal of Heuristics, vol. 8, no. 5, pp. 541-564.

Talukdar, S. (1992). A-teams for real-time operations. International Journal of Electrical Power and Energy Systems, vol. 14, no. 2-3, pp. 138-143.

Talukdar, S.N. (1998). Autonomous cyber agents: Rules for collaboration and concurrency. Proceedings of the Hawaii International Conference on System Sciences. Big Island, HI, USA: Institute of Electrical and Electronics Engineers Computer Society, pp. 57-61.

Talukdar, S.N.; Baerentzen, L.; Gove, A.; De Souza, P. (1998). Asynchronous Teams: Cooperation Schemes for Autonomous Agents. Journal of Heuristics, vol. 4, no. 4, pp. 295-321.