Alcom-FT Technical Report Series
ALCOMFT-TR-02-120

MALLBA: A library of skeletons for combinatorial optimisation*

E. Alba[3] F. Almeida[2] M. Blesa[1] J. Cabeza[2] C. Cotta[3] M. Díaz[3]
I. Dorta[2] J. Gabarró[1] C. León[2] J. Luna[2] L. Moreno[2] C. Pablos[2]
J. Petit[1] A. Rojas[2] F. Xhafa[1]

[1] LSI – UPC. Campus Nord C6. 08034 Barcelona (Spain).
[2] EIOC – ULL. Edificio Física/Matemáticas. 38271 La Laguna (Spain).
[3] LCC – UMA. E.T.S.I. Informática. Campus de Teatinos. 29071 Málaga (Spain).

Abstract. The MALLBA project tackles the resolution of combinatorial optimization problems using algorithmic skeletons implemented in C++. MALLBA offers three families of generic resolution methods: exact, heuristic and hybrid. Moreover, for each resolution method, MALLBA provides three different implementations: sequential, parallel for local area networks, and parallel for wide area networks (currently under development). This paper explains the architecture of the MALLBA library, presents some of its skeletons, and offers several computational results to show the viability of the approach.

1 Introduction

Combinatorial optimization problems arise in various fields such as control theory, operations research, biology, and computer science. Several tools offering parallel implementations for generic optimization techniques such as Simulated Annealing, Branch and Bound or Genetic Algorithms have been proposed in the past (see, e.g. [4, 7–9]). Some existing frameworks, such as *Local++*, its successor *EasyLocal++* [3], *Bob++* [2], and the IBM COIN open source project [6] provide sequential and parallel generic implementations for several exact, heuristic and hybrid methods, but lack features to integrate them.

The MALLBA project is an effort to develop an integrated library of skeletons for combinatorial optimization (including exact, heuristic and hybrid methods) dealing with parallelism in a user-friendly and, at the same time, efficient manner. Its three target environments are sequential computers, LANs of workstations and WANs. The main features of MALLBA are: integration of all the skeletons under the same design principles, facility to switch from sequential to parallel optimization engines, and cooperation among engines to provide more powerful hybrid skeletons, ready to use on commodity machines. Clusters of PCs

* http://www.lsi.upc.es/~mallba. Work partially supported by: Spanish CICYT TIC-1999-0754 (MALLBA), EU IST program IST-2001-33116 (FLAGS), Future and Emerging Technologies of EU contract IST-1999-14186 (ALCOM-FT) and Canary Goverment Project PI/2000-60. C. León partially supported by TRACS program at EPCC. M. Blesa partially supported by Catalan 2001FI-00659 pre-doctoral grant.

under Linux are currently supported and the software architecture is flexible and extensible (new skeletons can easily be added, alternative communication layers can be used, etc.).

In MALLBA, each resolution method is encapsulated into a skeleton. At present, the following skeletons are available: Divide and Conquer (DC), Branch and Bound (BnB), Dynamic Programming (DP), Hill Climbing, Metropolis, Simulated Annealing (SA), Tabu Search (TS), Genetic Algorithms (GA) and Memetic Algorithms. Moreover hybrid techniques have been implemented combining the previous skeletons, e.g., GA+TS, GA+SA, BnB+SA.

2 The MALLBA Architecture

MALLBA skeletons are based on the separation of two concepts: the concrete problem to be solved and the general resolution method to be used. While the particular features related to the problem must be given by the user, the main method and the knowledge to parallelize the execution of the resolution method is implemented in the skeleton. The users do not need to deal neither with the algorithmic part of the method nor with any parallelism issues.

Skeletons are implemented by a set of *required* and *provided* C++ classes which represent object abstractions of the entities participating in the resolution method. The *provided* classes implement internal aspects of the skeleton in a problem-independent way. The *required* classes specify information and behavior related to the problem. This conceptual separation allows us to define required classes with a fixed interface but without any implementation, so that provided classes can use required classes in a generic way. Fig. 1 depicts this architecture.

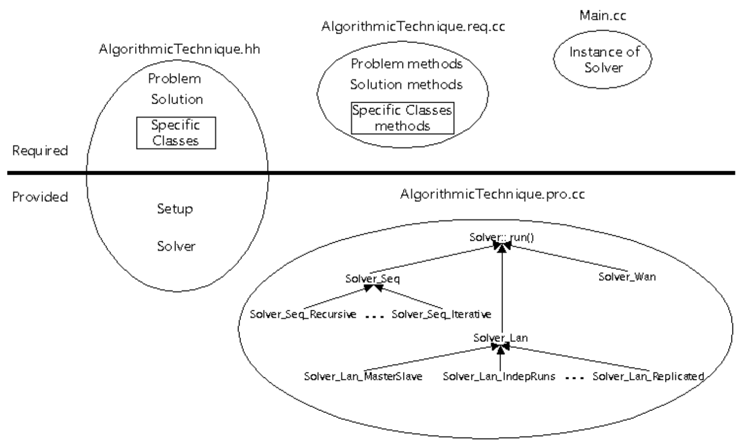

Fig. 1. Architecture of a MALLBA skeleton

More specifically, each skeleton includes the `Problem` and `Solution` required classes, that encapsulate the problem-dependent entities needed by the resolution method. The `Problem` class abstracts the features of the problem that are relevant to the selected optimization method. The `Solution` class abstracts the features of the feasible solutions that are relevant to the selected resolution method. Depending on the skeleton, other auxiliary classes may be required. On the other hand, each skeleton offers two provided classes: `Solver` and `Setup`. The former abstracts the selected resolution method. The later contains the setup parameters needed to perform and tune the execution. The `Solver` class provides methods to run the skeleton and also to consult or change its state dynamically. The only information the solver needs is an instance of the problem to solve and the setup parameters. In order to enable an skeleton to have different solver engines, the `Solver` class defines a unique interface and provides several subclasses that provide different sequential and parallel implementations (see Fig. 1).

Table 1. Results for the Resource Allocation Problem using the DP skeleton, over a network of 13 PCs (4 AMD K6 700 MHz and 9 AMD K6 500 MHz) connected through a Fast Ethernet network.

Stages-States	Sequential time (s) on fastest machine	Speed-up			
		2 procs. 700 MHz	4 procs. 700 MHz	4 procs. 700 MHz 4 procs. 500 MHz	4 procs. 700 MHz 9 procs. 500 MHz
1000-2000	457.79	1.97	3.92	4.12	6.01
1000-2500	714.87	1.98	3.94	4.30	6.02
1000-4000	1828.22	1.99	3.96	4.31	6.41
1000-5000	2854.04	1.99	3.97	4.24	6.42
1000-7000	5594.74	1.99	3.97	4.22	6.41
1000-10000	11422.60	1.98	3.97	4.18	6.38

Table 2. Results from IRSS and MSNP over a network of 9 AMD K6-2 450 MHz conected through a Fast Ethernet network. Maximum execution time fixed to 900s. An instance name like OR5x250-00 is an instance of 5 constraints and 250 variables. Averages calculated over 100 executions.

instance	best cost known	avg. deviation % from the best known					
		IR with Strategies			MS with Neighb. Part.		
		2 proc.	4 proc.	8 proc.	2 proc.	4 proc.	8 proc.
OR5x250-00	59312	0.028	0.015	0.020	0.051	0.024	0.020
OR5x250-29	154662	0.005	0.006	0.003	0.012	0.007	0.006
OR10x250-00	59187	0.045	0.047	0.045	0.079	0.064	0.064
OR10x250-29	149704	0.012	0.010	0.004	0.012	0.012	0.009
OR30x250-00	56693	0.023	0.022	0.017	0.041	0.028	0.030
OR30x250-29	149572	0.009	0.010	0.003	0.016	0.009	0.007

Table 3. Results for Maximum Cut. The experiments are done using 6 PCs (Pentium III, 700 MHz, 128 Mb) connected through a Fast Ethernet network. The "iterations" and "time" columns refer to the mean values of successful runs (those in which an optimal solution is found).

	Cut20-0.9			Cut100		
Algorithm	iterations	time(s)	successful runs	iterations	time (s)	successful runs
SA1	1742	0.11	92%	4048	3.96	78%
SA2	856	0.08	100%	2660	2.66	20%
GA	36	0.36	86%	399	78.66	6%
GASA1	5	0.50	100%	54	104.54	36%
GASA2	56	0.59	100%	178	43.84	4%
GASA3	71	0.73	94%	171	33.65	8%

3 Parallel Implementations

The skeletons of the MALLBA library are currently implemented for two target environments: sequential and LAN. The user is able to use different parallelization engines just by easily extending the sequential instantiations. These different implementations can be obtained by creating separate subclasses of the `Solver` abstract class (see Fig. 1). At present, we are using our own middleware layer `NetStream` implemented on top of MPI to ease communications.

3.1 Exact Methods. The MALLBA library follows Ibaraki's discrete Dynamic Programming (DP) approach for Multistage Problems to represent DP problems and the general parallelization scheme described in [5]. The parallelization performs a cyclic mapping of a pipeline on a ring topology. Divide and Conquer (DC) and Branch and Bound (BnB) have been parallelized using a master-slave strategy. While a queue of tasks suffices for the BnB, the DC requires of a hierarchy of queues. This structure gives support to the required synchronizations, since the corresponding combination phase has to occur after all the children have been solved. Factors to take into account are the relationship between the number of available processors and the number of generated subproblems, the depth of the generated subproblems and the communication, and computation capabilities of the hosting computer. The user can choose among several strategies provided by the skeletons. Table 1 shows the speedup obtained from an instantiation of the Dynamic Programming skeleton for the Resource Allocation Problem. The parallel engine shows a good scalability until four processors. Between four and eight processors performance decreases due to the slower machines introduced, but it remains increasing when introducing more processors.

3.2 Heuristic Methods: Tabu Search (TS). Fundamental ideas to design parallel strategies for meta-heuristics are already well-known [1]. The parallel implementations for TS in the MALLBA library include: *Independent Runs (IR)*, *Independent Runs with Search Strategies (IRSS)*, *Master-Slave (MS)* and *Master-Slave with Neighborhood Partition (MSNP)*.

We give in Table 2 some computational results obtained from the IRSS and the MSNP for the 0-1 Multidimensional Knapsack problem, for instances from the standard OR-Library. In the IRSS the communication time is almost irrelevant while in the MSNP there is a considerable communication time. This explains the fact that, for some instances, the solution found by IRSS is better than the one found by the MSNP.

3.3 Hybrid Methods: Genetic Annealing. The software architecture of MALLBA has allowed us a fast prototyping of several parallel hybrid skeletons: (GA) a parallel Genetic Algorithm with collaboration among sub-algorithms, (SA1) a parallel Simulated Annealing without collaboration among elementary SA's, (SA2) a parallel SA with collaboration among elementary SA's, and (GASA1) a hybrid skeleton where parallel GA's are applying SA as a local search operator inside their sequential main loop. Finally, we implemented two other hybrid skeletons where parallel GA's are run under different populations of solutions and then parallel SA's are applied, each one selecting some strings drawn from the final GA's population to improve them (by tournament -GASA2- or randomly -GASA3-). The construction of these hybrid algorithms has been extremely easy by using the MALLBA architecture.

We have solved the Maximum Cut problem: a high edge-density 20-vertex instance (`cut20-0.9`) and a 100-vertex instance (`cut100`) (see Table 3). In the hybrid parallel skeletons, SA is used at each reproduction event with probability 0.1 for 100 iterations. In the parallel-cooperation modes (all but SA1), the algorithms are arranged in ring topology (i.e., not master-slave), asynchronously migrating individuals each 200 iterations. Regarding the 100-vertex graph, the best result is provided by SA1 -4 sec.-, a parallel SA without cooperation (78% success in finding the optimum). The cooperative parallel version (SA2) is faster yet less effective (just 20% success); the reason for this result strives in that there exists a large number of local optima that SA2 repeatedly visits, while SA1 (without collaboration) focuses the search faster to an optimum in every parallel sub-algorithm, thus avoiding this "oscillation" effect and visiting a larger number of different search regions. The model GASA1 achieves an intermediate 36%-success value for `cut100`, but it is much more computationally expensive.

4 Concluding Remarks and Future Work

We have sketched the architecture of the MALLBA library. Also, we have given some computational results obtained over a cluster of heterogeneous PCs using Linux. Our results indicate that: skeletons can be instantiated for a large number of problems, sequential instantiations provided by users are ready to use in parallel, parallel implementations are scalable, general heuristic skeletons can provide solutions whose quality is comparable to *ad hoc* implementations for concrete problems, and the architecture supports easy construction of powerful hybrid algorithms. Our future work will focus on new skeletons for WAN.

References

1. T. Crainic, M. Toulouse, and M. Gendreau. Towards a taxonomy of parallel tabu search heuristics. *INFORMS Journal on Computing*, 9(1):61–72, 1997.
2. B. L. Cun. Bob++ library illustrated by VRP. In *European Operational Research Conference (EURO'2001)*, page 157, Rotterdam, 2001.
3. L. Di Gaspero and A. Schaerf. EasyLocal++: an object-oriented framework for the flexible design of local search algorithms and metaheuristics. In *4th Metaheuristics International Conference (MIC'2001)*, pages 287–292, 2001.
4. J. Eckstein, C. A. Phillips, and W. E. Hart. Pico: An object-oriented framework for parallel branch and bound. Technical report, RUTCOR, 2000.
5. D. González, F. Almeida, J. Roda, and C. Rodríguez. From the theory to the tools: Parallel dynamic progr. *Concurrency: Practice and Experience*, (12):21–34, 2000.
6. IBM. COIN: Common Optimization INterface for operations research, 2000. http://oss.software.ibm.com/developerworks/opensource/coin/index.html.
7. K. Klohs. Parallel simulated annealing library. http://www.uni-paderborn.de/fachbereich/AG/monien/SOFTWARE/PARSA/, 1998.
8. D. Levine. PGAPack, parallel genetic algorithm library. http://www.mcs.anl.gov/pgapack.html, 1996.
9. S. Tschöke and T. Polzer. Portable parallel branch-and-bound library, 1997. http://www.uni-paderborn.de/cs/ag-monien/SOFTWARE/PPBB/introduction.html.