Parallel noising methods embedded in an adaptive memory

Maxence Laurent*, Éric D. Taillard*, Michel Toulouse†, Teo Crainic†

*HEIG-Vd, University of Applied Sciences of Western Switzerland
Route de Cheseaux 1, Case Postale, CH-1401 Yverdon-les-Bains
{Maxence.Laurent,Eric.Taillard} at heig-vd.ch

†CIRRELT, Université de Montréal
C.P. 6128, Succursale Centre-Ville, Montreal H3C 3J7, Canada
{Michel.Toulouse, theo} at crt.umontreal.ca

This paper presents a high-level technique for parallelizing noising methods. To illustrate the technique, it is applied to the traveling salesman problem (TSP). It mixes 3 concepts proposed by various authors: 1) **Noising methods** proposed by Charon and Hudry [4]. The basic idea of noising methods is to add a random noise either to the problem data or to the objective function. 2) **Adaptive memory methods** of Taillard et al. [5, 6] that unify several metaheuristic concepts. Memetic algorithms, ant colonies hybridized with a local search, scatter search, vocabulary building and path relinking methods are analyzed with an adaptive memory point of view in [3]. So far, noising methods have not been embedded in the adaptive memory frame. 3) **Guided cooperative search** of Le Bouthillier, Crainic and Kropf [1, 2] who propose a special mechanism for exploiting an adaptive memory: The idea of [1, 2] is to try to get more pertinent information from the solutions stored in the memory, and especially from the bad ones. Mixing these 3 techniques together has been done as follows:

Memory: The memory (or data warehouse) is constituted of solutions. Each solution is classed into 3 categories : elite, intermediate and bad solutions. Each solution contains several components (for the TSP, a component is an edge: a route from one city to the next one). So, each component can be classified into 0, 1 or several of the 3 classes (no solution of the memory contains a given component, a component belongs to solutions of a single class or a component belongs to solutions of several classes). **Memory initialization**: The memory is initialized with solutions created with the the *Quick-Boruvka* procedure, as implemented in the *Concorde* software. These solutions are improved with the *Chained Lin-Kernighan* (CLK) procedure implemented in the *Concorde* software. **Noising**: Before launching CLK, the length of each edge is perturbed by a value that depends on a value $1 > r > 0$. This value r linearly decreases with the iteration number. For perturbing the length of the edges, we first multiply this length by a factor uniformly distributed between $1 - r$ and $1 + r$. If the edge only belongs to elite solutions, the perturbed length is then diminished by a factor $1 - r$. If the edge only belongs to bad solutions, the perturbed length is then increased by a factor $1 + r$. **Building a new solution**: The quickest way to build a new solution is to start from a solution already in memory. So, we took the best solution stored in memory. Since the length of all the edges is modified, the last is no longer a local optimum. The solution is improved with CLK.

Parallel implementation: The most demanding part of the algorithm is the optimization of solutions with the CLK procedure. This procedure is called several times with different data that are perturbed independently. So, it is easy to parallelize the method by running several CLK process on different processors. A separate process manages the memory, sends perturbed data to CLK processes and collects improved solutions from them. The workload of this process is relatively light compared to the workload of a CLK process. So, an asynchronous parallelization works very well, as soon as the number of processors does not exceed few dozen.

Preliminary numerical results The parallel noising method was tested with 5 optimization processes on the "National TSP collection" dataset[1]. The size of the instances varies from 29 to 71009 cities. For this problem set, we have obtained solutions that are on the average 0.1% above the best solutions known. None of the best solution value previously published have been improved, but our computational experiments were relatively limited (few CPU-days). By starting with an initial noise rate that is too low, the solutions contained in the memory are not sufficiently diversified and the method does not produces solutions significantly better than iterated CLK alone. The method was also tested on the Euclidean TSPLIB instances on a computer with 2 processors Xeon 3.2GHz and a shared memory (but still with 5 optimization processes). Due to memory limitation (each process stores its own data structure), few of the largest instances were not considered. For these instances the solution quality (measured in % above best solution value known) is provided in Figure 1 for both the parallel nosing method and CLK. The number of iteration of CLK was set to the total number of CLK iteration performed by the noising method. So, the computational effort is the same for both methods, but the time for the noising method is lower due to parallelization. We see in this figure that both methods provides excellent solutions. The average solution values is 0.053% above best known for parallel nosing and 0.058% for iterated LK. The sequential computational time for iterated LK is provided in Figure 2. The time seems to be proportional to $n^{1.12}$.

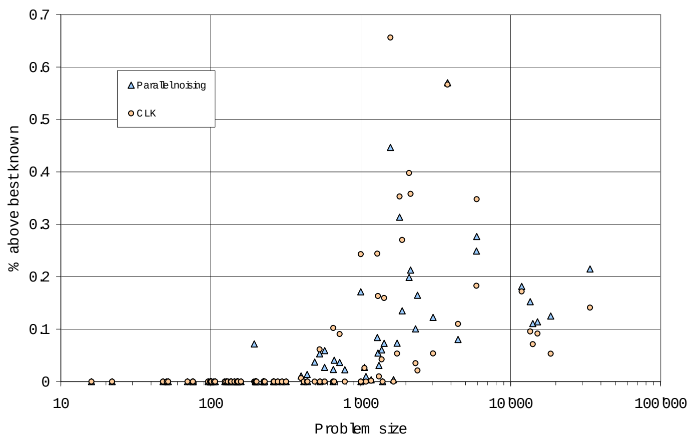

Figure 1: Solution quality of iterated LK and parallel noising for TSPLIB instances

[1]http://www.tsp.gatech.edu/index.html

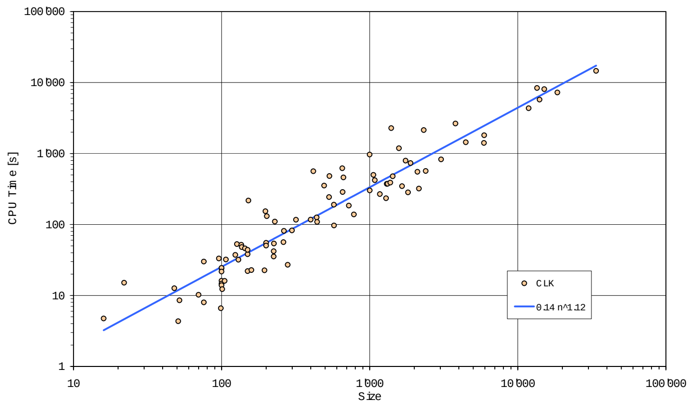

Figure 2: Computational time of iterated LK for TSPLIB instances

References

[1] A. Le Bouthillier, T. G. Crainic, P. Kropf (2005a). A guided cooperative search for the vehicle routing problem with time windows. *IEEE Intelligent Systems*, **20**(4), 36–42.

[2] A. Le Bouthillier, T. G. Crainic (2005b). A cooperative parallel meta-heuristic for the vehicle routing problem with time windows. *Computers and Operations Research*, **32**(7), 1685–1708.

[3] Dréo, J., Pétrowski, A., Siarry, P., & Taillard, E. D. (2006). *Metaheuristics for Hard Optimization: Methods and Case Studies.* Springer.

[4] I. Charon, O. Hudry (2001). The noising method: A generalization of some metaheuristics. *European Journal of Operational Research*, **135**, 86–101.

[5] Taillard, E. D. (1998). *Programmation à mémoire adaptative et algorithmes pseudo-gloutons : nouvelles perspectives pour les méta-heuristiques.* Thèse d'habilitation à diriger des recherches, Université de Versailles, France.

[6] Taillard, E. D., Gambardella, L. M., Gendreau, M., & Potvin, J.-Y. (2001). Adaptive memory programming : A unified view of meta-heuristics. *European Journal of Operational Research*, **135**(1), 1–16.