

Flexible cooperation in parallel local search (extended abstract)

Danny Munera, Daniel Diaz, Salvador Abreu, Philippe Codognet

▶ **To cite this version:**

Danny Munera, Daniel Diaz, Salvador Abreu, Philippe Codognet. Flexible cooperation in parallel local search (extended abstract). ACM Symposium on Applied Computing (SAC), Mar 2014, Gyeongju, South Korea. pp.1360 - 1361, 10.1145/2554850.2555140 . hal-01117539

HAL Id: hal-01117539

https://paris1.hal.science/hal-01117539

Submitted on 17 Feb 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Flexible Cooperation in Parallel Local Search

[Extended Abstract]

Danny Munera
University of Paris 1
France
Danny.Munera@univ-paris1.fr

Daniel Diaz
University of Paris 1
France
Daniel.Diaz@univ-paris1.fr

Salvador Abreu
University of Évora / CENTRIA
Portugal
spa@di.uevora.pt

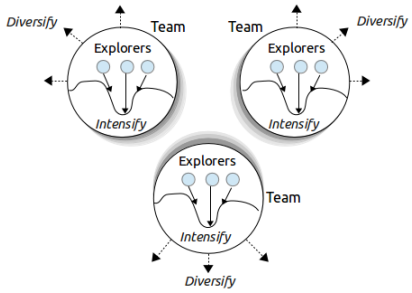
Philippe Codognet
JFLI - CNRS / UPMC /
University of Tokyo, Japan
codognet@is.s.u-tokyo.ac.jp

1. INTRODUCTION

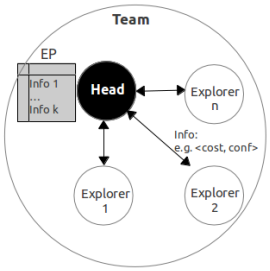
Constraint-Based Local Search (CBLS) consist in using Local Search methods [4] for solving Constraint Satisfaction Problems (CSP). In order to further improve the performance of Local Search, one possible option is to take advantage of the increasing availability of parallel computational resources. Parallel implementation of local search meta-heuristics has been studied since the early 90's, when multiprocessor machines started to become widely available, see [6]. One usually distinguishes between single-walk and multiple-walk methods: Single-walk methods consist in using parallelism inside a single search process, e.g. for parallelizing the exploration of the neighborhood, while multiple-walk methods (also called multi-start methods) consist in developing concurrent explorations of the search space, either independently (IW) or cooperatively (CW) with some communication between concurrent processes. Although good results can be achieved just with IW [1], a more sophisticated paradigm featuring *cooperation* between independent walks should bring better performance. We thus propose a general framework for cooperative search, which defines a flexible and parametric strategy based on the cooperative multi-walk (CW) scheme. The framework is oriented towards distributed architectures based on clusters of nodes, with the notion of "teams" running on nodes which group several individual search engines (e.g. multicore nodes). The idea is that teams are distributed and thus have limited *inter-node* communication. This framework allows the programmer to define aspects such as the degree of *intensification* and *diversification* present in the parallel search process. A good trade-off is essential to reach high performance. A preliminary implementation of the general CW framework has been done in the X10 programming language [5], and performance evaluation over a set of well-known benchmark CSPs shows that CW consistently outperforms IW.

2. FRAMEWORK DESIGN

We present a flexible, highly parametric cooperative search framework. This architecture allows the user to define a custom trade-off between intensification and diversification in the search process.



The figure above shows that all available solvers (*Explorers*) are grouped into *Teams*. Each team implements a mechanism to ensure intensification in the search space. Simultaneously, the teams collectively provide diversification for the search. Different teams will work on different regions of the search space. Inter-team communication is needed to ensure diversification while intra-team communication is needed for intensification.



As shown above, a team is composed of one *head node* and several *explorer nodes*, each one being an instance of a Local Search method. Each explorer periodically reports to the head node information on its search process (e.g. its current configuration, the associated cost, the number of iterations reached, the number of local minimum reached, etc.) The *head node* then makes decisions to ensure intensification in the search. To do so, it stores the configurations with the best costs in its *Elite Pool* (EP) and provides it, on demand, to explorers. Management of the elite pool has the potential for several strategies.

Periodically, each explorer asks the head node for an elite configuration. The head node may follow different strate-

gies in answering this request. Upon receiving the elite configuration, the worker node may, in turn, opt for different strategies in deciding what to do: to ignore or adopt it. The framework can be customized thanks to several parameters:

explorers per team: this number controls the trade-off between intensification and diversification.

team comm. topology: *all-to-all* (each team communicates with all other teams), *ring* (each team only shares information with its "adjacent" teams) or *random* (two teams are selected randomly to communicate each other).

team comm. interval: periodicity of the communication.

distance function: function used to check the closeness of two teams (to detect too close teams and diversify).

corrective action: action to perform when two teams are too close (e.g. the "worst" team can decide to clear or reset part of its internal state to move to another location).

elite pool size: the size of the set of elite configurations.

elite pool insert policy: defines the strategy of the head node when receiving a new configuration to insert in the EP. E.g. only insert it if it is not already present and if its cost is better than the worst already present.

elite pool request policy: defines the strategy of the head node when it receives a request for a configuration from the EP. E.g. pick a random configuration from the pool.

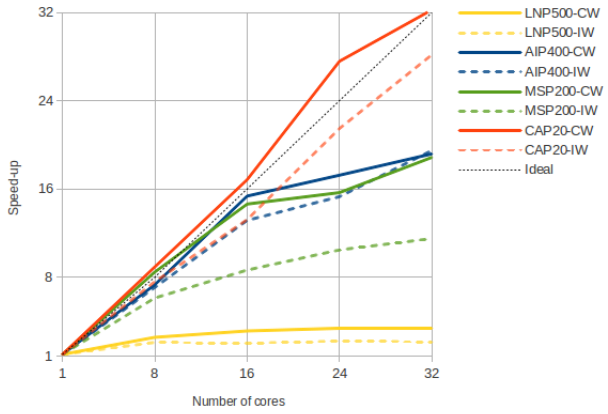
report interval: how frequently an explorer node communicates information to its head node.

update interval: how frequently an explorer node requests an EP configuration from its head node.

continuation policy: defines the strategy followed by explorers when receiving a configuration from the EP. E.g. if the received configuration is better than the current one, whether the explorer node switches to the EP configuration.

3. RESULTS AND ANALYSIS

We have implemented a prototype using the X10 parallel programming language. All explorers implement an instance of Adaptive Search [2, 3] specialized for permutation problems. To assess the performance of our Cooperative Walks (CW) implementation we compared it to an Independent Multi-Walks (IW) version, which runs several similar and isolated solvers in parallel (without any communication). We tested both versions on a set of 4 classic benchmarks: All-Interval Problem (AIP), Langford's Numbers Problem (LNP), Magic Square Problem (MSP) and the Costas Array Problem(CAP). For all cases, we ran 100 samples and averaged the times. The testing environment is a mixed cluster with 5 AMD nodes and 3 Intel nodes.



The above figure compares, for each benchmark, the speed-ups obtained with IW (dotted line) and CW (continuous line). In most cases, CW is getting closer to the "ideal" speedup. For instance, in CAP the cooperative strategy actually reaches super linear speed-ups over the entire range of cores (speed-up of 32.2 with 32 cores). The best gain reaches 69% in the MSP.

4. CONCLUSION

We have proposed a general framework for cooperative local search. It entails structuring the workers as teams, each with the mission of intensifying the search in a particular region of the search space. The teams are then expected to communicate among themselves to promote search diversification. The concepts and entities involved are all subject to parametric control (e.g., trade-off between intensification and diversification, the team communication topology, ...)

Initial experimentation with an early prototype has clearly proved that CW is able to systemtatically outperform the independent Multi-Walks parallel approach, even with very incomplete parameter tuning.

5. REFERENCES

[1] Y. Caniou, P. Codognet, D. Diaz, and S. Abreu. Experiments in Parallel Constraint-Based Local Search. In *proceedings of EvoCOP11*, pages 96–107, Torino, Italy, 2011. Springer.

[2] Philippe Codognet and Daniel Diaz. Yet another local search method for constraint solving. In *Stochastic Algorithms: Foundations and Applications*, pages 342–344. Springer, Berlin, 2001.

[3] Philippe Codognet and Daniel Diaz. An Efficient Library for Solving CSP with Local Search. In *5th international Conference on Metaheuristics*, pages 1–6, Kyoto, Japan, 2003.

[4] T. Gonzalez, editor. *Handbook of Approximation Algorithms and Metaheuristics*. Chapman and Hall / CRC, 2007.

[5] Vijay Saraswat, Bard Bloom, Igor Peshansky, Olivier Tardieu, and David Grove. X10 language specification - Version 2.3. Technical report, 2012.

[6] M.G.A Verhoeven and E.H.L. Aarts. Parallel local search. *Journal of Heuristics*, 1(1):43–65, 1995.