Self-Organization in Cooperative Tabu Search Algorithms

Michel Toulouse
School of Computer Science, University of Oklahoma
Email: toulouse@cs.ou.edu
Teodor Gabriel Crainic
Centre de Recherche sur les Transports, Université de Montréal
Email: theo@crt.umontreal.ca
Brunilde Sansó
École Polytechnique, Université de Montréal
Email: bruni@crt.umontreal.ca
K. Thulasiraman
School of Computer Science, University of Oklahoma
Email: thulasi@cs.ou.edu

Abstract— The search history of memory based heuristics like tabu search can be used to design a category of parallel algorithms called cooperative search. These algorithms execute in parallel several search programs on the same optimization problem instance. At run time, the data gathered in the memory by one sequential search program need not be used only by this program, but it can be recycled and shared with other concurrently executing tabu search programs for the same purpose. Several such cooperative search algorithms have already been proposed in the literature, but yet, very little is known about how they work as problem solving methods, particularly about their optimization properties. In this paper we compare the global behavior of cooperating and non-cooperating tabu search programs. We show that cooperating programs tend to have a search pattern which is less diversified than non-cooperating programs. Our findings also indicate that this second order impact of the sharing of gathered data on the search behaviors of cooperating programs is not related to the optimization properties of the individual tabu programs.

I. INTRODUCTION

Memory based local search methods such as the tabu search method use data gathered from previously visited regions of the solution space as input of their acceptance criteria to select new configurations. In a parallel programming environment, there seems to be an obvious benefit for a population of concurrently executing tabu search programs to reuse the data which have being gathered in the tabu memories of a program to improve the acceptance criteria of the other concurrently executing tabu programs. For the past 10 years, several cooperative algorithms have been designed and implemented based on the sharing of gathered information, using tabu search [4] or other search methods [2], [6], [8], [9], [10], [12]. Most empirical studies conclude that a population of search programs exchanging gathered data obtains better performance compared with the same population of non-interacting programs. But still, according to several researchers [1], [5], [11], we lack a theory that could explain and predict how the exchange of information improves the performance of cooperating programs. In the present paper we address a particular aspect of this problem

0-7803-4778-1/98 $10.00 © 1998 IEEE

by showing that some of the modifications occurring in the search pattern of cooperating tabu programs are not related to the content of the messages which are exchanged.

Our demonstration is based on an experiment where data gathered in the tabu memories are replaced by data provided by the state of a cellular automaton (CA) evolving in a complex non-periodic attractor. Such data are meaningless in terms of information that can be used by the tabu search programs, but these data modify the sequence of configurations visited by the "cooperating" tabu search programs. By comparing the *diversity* of the configurations generated by a population of programs exchanging data from their tabu memories and the same population of programs interacting through the data provided by a cellular automaton, we show that the diversity of the configurations of these two populations is very similar, but much lower than the same population of programs which do not interact with each other in any manner. Other tests which are based on randomly generated data show diversity values which are higher than cooperating programs using (reusing) attributes of previously visited configurations. We conclude that, besides being useful as input to the accepting criteria of the local search methods, the interactions among search programs create a second order phenomenon which is a kind of self-organization of the search patterns of these programs. This self-organization behavior is not related to the optimization properties of the search programs since it can be achieved by sharing data which are totally insignificant for the search methods.

This paper is organized as follows. In the next Section we describe the issues related to local search methods and to the design of our cooperative tabu search method. In Section 3 we introduce the parallel procedures and a few results from the cellular automata theory on which depends the experimental part of this study. In Section 4, we present and analyze the numerical results from our experiments and in Section 5 we conclude.

II. COOPERATIVE TABU SEARCH METHOD

Local search methods are among the approximation methods most often used to find near optimal solutions to NP-hard combinatorial optimization problems. A local search starts with an initial configuration and generates a sequence of configurations by perturbing the current configuration with a transition move. New configurations are generated with the application of transition moves, they are accepted as configurations in the sequence of configurations if they meet some *acceptance criteria.* Acceptance criteria are essential components of local search methods. We can used them to characterize search methods. For example, tabu search methods [7] are based on sophisticated acceptance criteria which depend on the history of the exploration in the solution space. Attributes of previously generated configurations are stored in a multi-layered time related memory structure. Acceptance criteria then select configurations according to some correlations that attributes of these configurations have with attributes of configurations stored in the "tabu memories".

As the acceptance criteria become more sophisticated, the calibration of these criteria reaches a point where their impact on the performance of the search methods becomes a challenging issue. This is the case for search methods such tabu search, simulated annealing and genetic algorithms. In tabu search methods, *search parameters* such as which attributes to use, how much of the history should be stored, when a rule should overwrite another, etc., have to be calibrated before-hand or to evolve during the computation. Each of these parameters affects the precise sequence of configurations accepted for a given problem instance. On the other hand, the different configuration sequences resulting from different calibrations provide quite an opportunity for using parallel computers to enhance the performance of a method like tabu search. Given that each calibration is susceptible to generate a different sequence of configurations, calibration

can then be used to induce different explorations of the solution space, each exploration becoming a task that can be executed concurrently with the others. Cooperative tabu search algorithms are based on this implicit decomposition of the solution space using different search parameters to obtain parallel tasks. Cooperation is then obtained by exchanging at run time some of the attributes stored in the tabu memories of the concurrent programs.

III. PARALLEL PROCEDURES

Let $P_0, P_1, \ldots, P_{p-1}$ be a population of p tabu search programs (the tabu search programs are similar to the one described in [3]), each program using a different calibration of the search parameters. Based on this population of search programs, we have designed five types of parallel procedures. The first one, *cooperative search* ([Proc. 1- CS] in Table I), consist of programs interacting by asynchronous exchanges of good configurations in their sequence. The second one, *independent searches* ([Proc. 2- IS] in Table I), is a parallel procedure where each search program generates a configuration sequence based only on its own set of search parameters, without interacting in any manner with the other programs. The third parallel procedure, *random data* ([Proc. 3- RD] in Table I), is based on randomly generated configurations which are synchronously sent to each program every time they have completed the generation of δ configurations.

The fourth and fifth parallel procedures, SIM_{18} and SIM_{22} ([Proc. 4- SIM_{18}] and [Proc. 5- SIM_{22}] in Table I), are synchronous cooperative algorithms which use cellular automata to create interactions among the programs. Each time a program has generated δ configurations, it sends one of these δ configurations to the CA. The configuration sent by a program P_i to the CA is a binary vector m_i of length n (the number of decision variables in the optimization problem). Once the CA has received p configurations (one from each program), it concatenates them into a single binary vector M of length

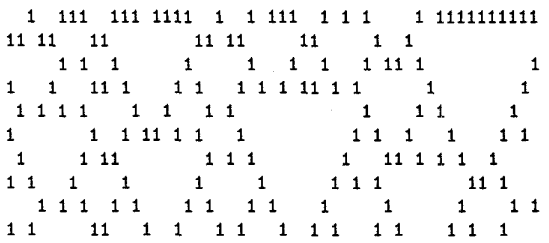

Fig. 1. Rule 18 from an initial state generated with p=1/2

$p \times n$ representting a state in the state space of the CA. The vector M is used as the initial state of the CA. Once initialized, the CA performs t iterations using a particular transition rule. Once the CA has entered its t-th state, the binary vector of this state is partitioned into p configurations, each one being returned to a different program. Each program then resumes its exploration of the solution space from the configuration received from the CA. This outer loop is repeated until the tabu search programs reach their respective stopping criteria.

The parallel procedures SIM_{18} and SIM_{22} are based on a cellular automaton which evolved according to transition rules 18 and 22 respectively (see [13]). The time evolution of both rules is known to induce complex non-periodic attractors in the state space of one dimensional cellular automata. In Figure 1 for example, the structures (triangles) show that the state of each cell depends increasingly on a larger number of cells relatively to the number of iterations of the cellular automaton. Assume that ϕ stands for either rule 18 or 22. We deduce a global transition rule Φ between the set of configurations:

$$\Phi : \Sigma \rightarrow \Sigma \qquad (1)$$

where Σ is the set of possible configurations of the cellular automaton. (Equation 1 means that the transition rule ϕ is applied to each configuration in Σ.) Let $\Omega^\tau = \Phi^\tau \Sigma$ be the set of configurations generated after τ iterations of Φ. For cellular automata transition rules which generate complex non-periodic attractors, the following

holds [14]:

$$\Omega^{\tau+1} = \Phi\Omega^{\tau} \subseteq \Omega^{\tau} \qquad (2)$$

i.e. the application of the rule Φ to all configurations of the set Ω^{τ} at iteration τ results in a set $\Omega^{\tau+1}$ at iteration $\tau+1$ where the number of configurations is lower or equal to the set Ω^{τ}. This evolution towards a lower number of configurations in the set Ω means that the entropy associated with the set Ω^{τ} decreases or stays the same as the number of iterations increases. Such decreases in the entropy is a typical manifestation of a self-organized behavior.

It can also be shown that the decrease in the number of configurations in the set Ω occurs because blocks of sub-configurations get excluded from the configurations generated by the rule ϕ of the cellular automaton:

Proposition 1: A sub-configuration of length l is excluded after τ iterations of a rule ϕ if there is no sub-configuration from a one dimensional cellular automaton initial configuration of length $(l+2)\times\tau$ which evolves toward the sub-configuration of length l.
(see [14] for details).

IV. EXPERIMENTAL RESULTS

Each of these five parallel procedures has been run on twelve instances of the location-allocation combinatorial optimization problem as introduced in [3]. Each problem instance is searched using 4, 8 and 16 sequential tabu search programs which are slightly different versions of the sequential TS method described in [3]. Each program executes 300 TS iterations in parallel on a network of Sparc workstations. The location-allocation problem has two types of decision variables: design variables, which are boolean (0/1); and flow variables, which are continuous. The tabu search method runs in the space defined by the design variables, therefore the solution space is given by $X^n \subseteq B^n$.

We have compared these five parallel procedures in term of the diversity of the sequence of configurations generated by the search programs. Let p be the number of search programs, t the number of iterations executed by each program. In terms of the number of configurations visited, a parallel procedure can reach $p \times t$ configurations with p sequential programs. The diversity is measured using the Hamming distance between the configurations in the following way:

- $H(m_i, m_j)$ is the Hamming distance between two configurations m_i and m_j from any programs in the population;
- $H^{i+} = \sum_{j=0}^{j=(t\times p)-1} H(m_i, m_j)$ is the Hamming distance between a configuration m_i and the $(t\times p)-1$ configurations of the p programs;
- $div = \sum_{i=0}^{i=(t\times p)-1} H^{i+}$ is the *global Hamming distance*, i.e. the Hamming distance between all the configurations of all the p programs.

Assuming that $div(Prob_i)$ is the diversity of a given parallel procedure for the problem instance $Prob_i$, then Table I shows the average diversity $((\sum_{i=1}^{i=12} div(Prob_i))/12)$. The diversity of the cooperative search is substantially lower than for the independent searches, the diversity of parallel procedures based on self-organized cellular automata is closer to cooperative search compared to parallel procedures based on randomly generated messages.

	Number of programs		
|| procedures	p = 4	p = 8	p = 16
Proc. 1- CS	8210583	33650795	137259828
Proc. 2- IS	10329786	39522603	161269919
Proc. 3- RD	8964296	35525870	147515899
Proc. 4- SIM_{18}	7670456	31167806	129597964
Proc. 5- SIM_{22}	8412034	32544760	137165346

TABLE I
AVERAGE DIVERSITY OF THE CONFIGURATIONS GENERATED BY 4, 8 AND 16 SEARCH PROGRAMS

A. Diversity of SIM_{18} and SIM_{22} procedures

The structuring process taking place in the states of a CA such as the one depicted in Figure 1, corresponds according to Equation (2) to a decreases in the entropy of the set of states that

can be reached by the CA at each iteration. As the entropy decreases, this translates according to Proposition (1) into a exclusion of some sub-configurations (blocks of length l) from the states reached by the CA. (For example, in the last configuration of Figure 1, the patterns [1101011], [110001011] and [1101001] are excluded, and they will never appear again in any state visited by this CA.)

The cellular automata of SIM_{18} and SIM_{22} return configurations to the tabu search programs similar to those appearing in Figure 1. The process of excluding sub-configuration patterns from occurring in the vector M is symmetric to a process repeating the same sub-configuration patterns (as more patterns are excluded, the likelihood that similar patterns will appear is increased). Consequently, as an increasingly larger number of sub-configuration patterns are excluded from the vector M generated by the CA, different search programs receive similar configurations from the CA. This is the way that the dynamics of the complex non-periodic attractors of rules 18 and 22 reduce the diversity of the parallel procedures SIM_{18} and SIM_{22}.

B. Diversity of the cooperative search procedures

In the cooperative search procedures, programs interact by swapping a configuration m_i between two programs p_i and p_j. Once m_i has been swapped from p_i to p_j, it becomes the current configuration of program p_j from which this program resumes the execution. At some point, program p_j will interact with a program p_k to which it will send a configuration m_j. The interactions based on configurations m_i and m_j are said to be correlated to each other and are part of a same diffusion process. Figure 2 shows a correlated sequence of interactions which interfere with the search method of the sequential programs. As we can see from this figure, once the configuration m_0 has been swapped in program p_1, the state of only a few variables (those with value "2" in the figure) will be changed by the

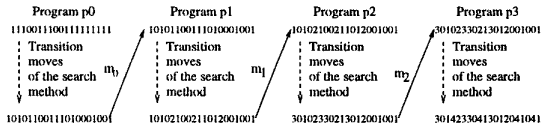

Fig. 2. Diffusion process among search programs

transition moves of program p_1. When configuration m_1 is swapped from p_1 to p_2, only a few variables have a different state in the two configurations m_0 and m_1. This means that the swapping of m_1 from p_1 to p_2 is equivalent to copying the state of several variables from program p_0 to p_2. The state of variables in a same diffusion process which are not changed by the transition moves of the programs are copied across several search programs. For example, in Figure 2 the first sub-configuration 010 between 3 and 2 in program p_2 has been copied from program p_0, to p_1, p_2 and p_3. The occurrence of the same blocks (like 010) in the configurations of several programs is the main factor contributing to the reduction of the diversity of cooperative search procedures.

C. Interpretation of the results

Attributes of good solutions are sent to other programs such that they can be reused to improve the performance of cooperating programs. This is the main objective in the design of cooperation among search programs. The results of the present study show that similar to the propagation of symbolic information (such as configurations selected according to the cost function), another type of information, the sub-configurations, are repeatedly copied across several search programs. Those sub-configurations are never evaluated by the cost function, they are just sent from one program to another using as medium the configurations selected by the cost function. Although they are not evaluated, the repeated occurrence of the sub-configurations has a strong impact on the search behavior of cooperating programs, as it shows-up in the diversity values of these parallel procedures.

The similarity of the diversity between cooperative search procedures and parallel procedures based on non-periodic dynamics of cellular automata, indicates that coordination among the cooperating programs is emerging from their interactions through the sharing of information. This coordination occurs spontaneously without being specifically introduced in the design of the cooperative algorithms, therefore we refer to this global behavior of cooperative procedures as a form of self-organization process. This global behavior is clearly not subordinated to the optimization logic of the search programs. Rather it is a second order phenomenon based on the correlated interactions (diffusion dynamics) among the search programs. Although self-organization is shaping the search behavior of cooperating programs, our study brings no evidence that it should be credited for automatically improving the performance of cooperative algorithms.

V. CONCLUSION

Memory based local search methods use the history of the solution process to determine the choice of the configurations while exploring the solution space of a problem instance. Many cooperative algorithms share the histories of the solution processes among search programs with the hope to find better solutions to the optimization problem. In this paper we have shown that the sharing of gathered data has a global impact of reducing the diversity of the configurations visited in the solution space by a population of search programs. We have also shown that this global behavior is independent of the optimization properties of the data exchanged. This kind of global phenomena plays a role in the performance of cooperating algorithms. We believe that by addressing this aspect of cooperative search algorithms, future designs can improve in an important manner their performance.

REFERENCES

[1] S.H. Clearwater, T. Hogg, and B.A. Huberman. Cooperative Problem Solving. In B.A. Huberman, editor, *Computation: The Micro and the Macro View*, pages 33–70. World Scientific, 1992.

[2] S.H. Clearwater, B.A. Huberman, and T. Hogg. Cooperative Solution of Constraint Satisfaction Problems. *Science*, 254:1181–1183, 1991.

[3] T.G. Crainic, M. Gendreau, P. Soriano, and M. Toulouse. A Tabu Search Procedure for Multicommodity Location/Allocation with Balancing Requirements. *Annals of Operations Research*, 41:359–383, 1993.

[4] T.G. Crainic, M. Toulouse, and M. Gendreau. Towards a Taxonomy of Parallel Tabu Search Algorithms. *INFORMS Journal of Computing*, 9(1):61–72, 1997.

[5] I.R. East and J. Rowe. Effects of Isolation in a Distributed Population Genetic Algorithm. *Lecture Notes in Computer Science*, 1141:408–419, 1996.

[6] B. Gendron and T.G. Crainic. Parallel Branch-and-Bound Algorithms: Survey and Synthesis. *Operations Research*, 42(6):1042–1066, 1994.

[7] F. Glover and M. Laguna. *Tabu Search.* Kluwer Academic Publishers, 1997.

[8] P. S. Laursen. Problem-Independent Parallel Simulated Annealing Using Selection and Migration. *Lecture Notes in Computer Science*, 866:408–417, 1994.

[9] B. Manderick and P. Spiessens. Fine-Grained Parallel Genetic Algorithm. In J. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 416–421. Morgan Kaufmann Publishers, 1989.

[10] P. Moscato and M.G. Norman. A 'Memetic' Approach for the Traveling Salesman Problem. Implementation of a Computational Ecology for Combinatorial Optimization on Message-Passing Systems,. In M. Valero, E. Onate, M. Jane, J.L. Larriba and B. Suarez, editors, *Parallel Computing and Transputer Applications*, pages 187–194. IOS Press, Amsterdam, 1992.

[11] C. Pettey and M. Leuze. A Theoretical Investigation of a Parallel Genetic Algorithm. In J. D. Schaffer, editor, *Proc. Third Int. Conference on Genetic Algorithms and their Applications*, pages 398–405. Morgan Kaufmann Publishers, 1989.

[12] C. Pettey, M. Leuze, and J. Grefenstette. A Parallel Genetic Algorithm. In J.J. Grefenstette, editor, *Proc. Second Int. Conference on Genetic Algorithms and their Applications*, pages 155–161. Lawrence Erlbaum Associates Publishers, 1987.

[13] S. Wolfram. Statistical Mechanics of Cellular Automata. *Reviews of Modern Physics*, 55(3):601–644, 1983.

[14] S. Wolfram. Computation Theory of Cellular Automata. *Comm. Math. Phys*, 96:15–57, 1984.