Global Optimization Properties of Parallel Cooperative Search Algorithms: A Simulation Study

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
K. Thulasiraman
School of Computer Science
University of Oklahoma
Email: thulasi@cs.ou.edu

Abstract

Cooperative search is a parallelization strategy where parallelism is obtained by concurrently executing several search programs for the same optimization problem instance. The programs cooperate by exchanging information on previously explored regions of the solution space. When the sharing of information overlaps among several programs, changes in the search behavior of one program can propagate over time to several other programs; this is a known process in physical systems called diffusion. The optimization properties of diffusion dynamics in cooperative algorithms have not been formally established. However, it is generally believed that when the selection of shared information is biased by the cost function, diffusion dynamics help to improve the search of cooperating programs. In this study, we simulate this aspect of cooperative algorithms using cellular automata (these are artificial dynamical systems often used to simulate the dynamics of complex systems). Our results show that the sharing of information based on the cost function does not affect the diffusion dynamics and therefore does not seem to help the optimization strategy of cooperating programs. However, this study increases our understanding of the role play by diffusion processes in cooperative algorithms. We suggests new approaches that can help to subordinate diffusion dynamics to the optimization goals of the search programs.

Keywords: Cooperative search algorithms, parallel metaheuristics, combinatorial optimization.

1 Introduction

We consider combinatorial optimization problems having the form

$$min\,(max)\{c(x)|x \in X\}$$

where X is a set of feasible solutions (configurations) and c is a cost function. Local search algorithms are commonly used to find near-optimal solutions to this kind of optimization problems. They are simple iterative improvement methods seeking a transition from a current configuration x in the problem's solution space to a configuration y in the neighborhood of x such that $c(y) < c(x)$. The choice of y in the neighborhood of x can be based on selection strategies such as: y is the first improving configuration or y is the best improving configuration. More complex searches such as *tabu search* (TS), *simulated annealing* (SA), *genetic algorithm* (GA) and others, use more sophisticated selection strategies often accepting non-improving transitions; they are therefore capable of a more extensive exploration.

Cooperative search is a parallelization strategy increasingly used for search methods. Procedures such as coarse-grained *parallel genetic algorithms* (PGA) (Pettey, Leuze and Grefenstette 1987), fine-grained PGA (Manderick and Spiessens 1989), memetic algorithms (Moscato and Norman 1992), parallel Branch-and-Bound (Gendron and Crainic 1994), parallel tabu search algorithms (Crainic, Toulouse and Gendreau 1995), parallel simulating annealing procedures (Laursen 1994), and special purpose artificial intelligence cooperative searches (Clearwater, Huberman and Hogg 1991) are examples of the application of this parallelization strategy to well known search heuristics. Contrary to conventional parallelization strategies, the parallel tasks of cooperative procedures are not obtained from the decomposition of the data or the search program. Most often, these tasks are obtained by executing concurrently several search programs using the same data set. These parallel tasks are said to cooperate with each other because, at run time, search programs exchange information on previously explored regions of the solution space.

Each sequential program explores differently the solution space and can hold unique data about configurations visited. By providing shared information among search programs, cooperative design aims to improve the performance of the individual search programs through the reuse of information collected by other programs. However, to be effective, shared information has to interfere somehow with the control structure (the transitions) of the sequential programs. It is not often clear how these interferences affect the optimization strategy of the search methods executed by each sequential programs (they could mislead the optimization strategy). Furthermore, these interferences have a cumulative impact on the search programs. The situation is similar to rule-based search systems in artificial intelligence. Once a rule is fired (information is shared) it changes the working memory (it causes a different configuration to be visited) which then affects the choice of the next rule to be fired (which information is going to be shared next), and so on. This propagation pattern is a well known process in physical systems and is referred to as *diffusion*. In most cooperative algorithms, the reused information is at the origin of sequences of correlated interactions among programs corresponding to diffusion processes. For example, the stepping-stone architecture of coarse-grained PGA is an explicit example where the overlap of the neighborhoods

gives birth to a diffusion of shared information among the sub-populations.

Currently, diffusion dynamics are not specifically addressed in the design of cooperative algorithms. Rather, the main concern is about finding cooperation (or competition) schemes, and some criteria to select which information should be reused. Selection criteria are defined considering the application of the cooperation scheme at a particular point in time, without considering the cumulative impact that the application of a given selection criterion can have on subsequent applications of a cooperation scheme.

Let identify transitions which are based on shared information as *meta-transitions* to differentiate them from the *search method transitions* originating uniquely from the control structure of the sequential search methods. Actually, in most cases, the criteria used to select the meta-transitions are based on the cost function of the optimization problem, e.g. the selected meta-transition is the one which minimizes (or maximizes) the cost $c(y)$ of the next configuration visited. It is generally assumed (most often implicitly) that this type of selection criteria can take care of the diffusion dynamics. Each search program executes its local optimization strategy on the same problem, but it is assumed that as a result of cooperation (which includes the cumulative impact of diffusion dynamics), the search programs develop *global optimization properties*, meaning that at least some of the local optimization strategies will be improved by cooperation (i.e. the exchange of information will not completely mislead cooperating programs by driving them to worst solutions than if there were no cooperation).

In the present paper we study the diffusion dynamics of information in cooperative algorithms. Because of the diffusion processes, cooperative algorithms are a special case of space-extended dynamical systems, similar to connectionnist models of computation like cellular automata (CAs) and some artificial neural networks (ANNs). We consider the possibility that the symbolic evaluation of meta-transitions based on the cost function might yield diffusion dynamics which are irrelevant from an optimization point of view. (As for CAs and ANNS, if the interactions are not well designed, local rules can induce dynamics which do not have any optimization properties.) Therefore, we compare the *diversity* of the configurations visited by a set of search programs which use different cooperation schemes: some based on the cost function, others on the diffusion dynamics of cellular automata, and some others having no diffusion dynamics. We use two measurements to estimate the diversity of the configurations: the Shannon entropy and the Global Hamming Distance (GHD) between configurations. The Shannon entropy measures diversity in terms of the number of different configurations versus the number of configurations visited, while the GHD measures the diversity in terms of different sub-blocks among the configurations visited.

Our results show that cooperation schemes have no significant impact on the diversity of the configurations in terms of the Shannon entropy. However the GHD values are significantly lower when the cooperation schemes involve diffusion processes. Diffusion dynamics change the search behavior of cooperating programs in terms of the frequency of identical sub-blocks appearing in different configurations (or the same configuration). Furthermore, when there are diffusion processes, the mode of evaluation of the meta-transitions has no impact on the diversity of the configurations according to both measurement methods. This observation establishes that the selection mode of the meta-transitions is not a relevant parameter affecting the diffusion dynamics of the cooperative procedures tested. In that case, if any global optimization properties exist

for those procedures, they are not triggered by the type of selection criteria applied to the meta-transitions. The cumulative impact of sharing information among search programs does not seem to be very helpful to the optimization goals of cooperating programs when meta-transitions are selected based on the cost function. We show how this situation can be changed and how we can transform cooperative algorithms into more efficient optimization methods.

The paper is organized as follows. In Section 2 we describe the methodology used in the present study. In Section 3 we provide some background on the cellular automata theory. In Section 4 we introduce the implementation details of the cooperation schemes based on cellular automata and on randomly generated meta-transitions. In Section 5 we describe the experimentations and in Section 6 we conclude.

2 Methodology

This study is based on experiments involving a same set of sequential tabu search (TS) programs which cooperate according to different cooperation schemes. We model cooperative algorithms based on TS as discrete-time dynamical systems (see the details of this model in Toulouse, Crainic and Sansó 1997). Assume ψ is a p-dimensional state vector, and f some iterative operator. A discrete-time dynamical system has the structure

$$\psi(k+1) = f(\psi(k)), \quad k = 0, 1, 2, \ldots, \tag{1}$$

where the state vector $\psi(k+1)$ depends only on vector $\psi(k)$ and the operator f (we assume the system has no input). If ψ and f are separable in p components, we can then rewrite (1) as

$$\psi_i(k+1) = f_i(\psi_0(k), \psi_2(k), \ldots, \psi_{p-1}(k)) \tag{2}$$

where $\psi_i(k)$ is the i-th component of $\psi(k)$, and f_i is the i-th component of operator f.

Let $p_0, p_1, \ldots, p_{p-1}$ be the set of sequential TS programs making the cooperative search procedure. We can put this procedure in a framework like (2) by taking program p_i as the component f_i of the operator $f = \bigcup_{i=0}^{i=p-1} p_i$. Each program p_i is a subsystem of the discrete-time dynamical system (each p_i can be further split in two different mappings d_i and u_i where d_i represents the transitions based on the search method and u_i represents the meta-transitions). Finally, without lost of generality, we can assume that the state vector $\psi(k)$ is given by the configurations in the solution space visited at iteration k by the set of of sequential TS programs $p_0, p_1, \ldots, p_{p-1}$. Therefore a configuration x at iteration k of a program p_i can be seen as the i-th component of the state vector $\psi(k)$.

If no information is shared among the p search programs, Equation (2) can be stated as $\psi_i(k+1) = f_i(\psi_i(k))$ for $i = 0, 1, \ldots, p-1$. This set of equations does not define a system because transitions depend only on the internal dynamics of each subsystem, namely the transitions produced by the search methods. However if information is shared, transitions occurring in a program p_i at iteration $k+1$ depend on several components ψ_j of the state vector ψ at iteration k. The vector $\psi_1(k), \psi_2(k), \ldots, \psi_p(k)$

of Equation (2) represents the inter-subsystems dynamics affecting the outcome of the transition of program p_i at iteration $k+1$. In this case, the set of tabu search programs is truly a dynamical system where the sequential programs are subsystems, the search path of program p_i is the trajectory in the state space of subsystem (ψ_i), the matrix of elements $\psi_{i,1}(k), \psi_{i,2}(k), \ldots, \psi_{i,p}(k),\ i = 0, \ldots, p-1$ represents the interactions among the sub-systems ($\psi(i,j)(k)$ is the interaction between programs p_i and p_j at iteration k) and the state vector ψ represents the trajectory of the dynamical system in its state space (defined as X^p, the Cartesian product of X with itself $p-1$ times).

A diffusion process in such a system occurs when the state of a subsystem p_i at iteration $k+1$ is determined by interactions between at least two other subsystems p_j and p_l at iteration k and $k-x,\ \ x < k$:

$$\psi_i(k+1) \quad = \quad p_i(\psi_j(k)) \tag{3}$$

and

$$\psi_j(k-x) \quad = \quad p_j(\psi_l(k-x-1)) \tag{4}$$

where the value of $\psi_j(k)$ in Equation (3) depends on the value of $\psi_j(k-x)$ in Equation (4). If the selection of meta-transitions is based on the cost function, then the meta-transition $p_j(\psi_{j,l}(k-x-1))$ in Equation (4) is said to be preferred over other meta-transitions (and possibly the local search transition $p_j(\psi_j(k-x-1))$) because it yields a configuration $\psi_j(k-x)$ for which $c(\psi_j(k-x))$ is lower than the configurations obtained by other transitions.

For diffusion processes such as the one portraited by Equations (3) and (4), where choices of meta-transitions at one iteration depend on choices made by other programs at previous iterations, it is expected that the cumulative impact of these choices would yield better solutions when the selection is based on the cost function. This is equivalent to say that the selection mode of the meta-transitions is a parameter controlling the diffusion dynamics. However, verifying this statement is not easy. The internal dynamics of each program is based on heuristics, and interactions among search programs often occur according to the internal state of the programs, therefore deriving any analytical model of a such system without making unduly strong assumptions is almost impossible. Comparing the cost of the best configurations visited by different cooperative algorithms will not help either. Even if we compare these configurations based on different selection criteria, this does not tell anything about the relation existing between selection criteria and diffusion dynamics.

Cooperative procedures seen as complex dynamical systems provide a more sensitive approach to analyze cooperative algorithms. Rather than running another empirical study of the behavior of cooperative algorithms (in place of an analytical study), we can run simulations of these algorithms. Simulations are a common approach to increase our understanding of complex systems. Furthermore, the usual statistical measurements from dynamical systems theory provide numbers to compare with and means to assess the accuracy of the simulations.

Place in the context of dynamical systems theory (Equation (1)), the statement above means that the selection mode of the meta-transitions using the cost function

can direct the trajectory of the system in its state space such that it will overlap with more interesting regions of the solution space. If the factors affecting the behavior of the heuristics at the level of each program are kept constant, this is only possible through the matrix of interactions among the subsystems. Given that we don't know of any model which expresses the correlations between the elements of this matrix and the heuristics, or between the elements themselves at different iterations, we rather simulate this matrix of interactions. Simulations are based on the definitions of new cooperation schemes, some capable to generate well characterized diffusion dynamics among the search programs.

We use two simulation models, one is based on randomly generated meta-transitions according to a normal distribution, and the other is based on the cellular automata theory. CAs are mathematical dynamical systems which have well characterized properties. We use these properties to control the interactions among search programs. We thus compare the search behavior of cooperative procedures with the following set of cooperation schemes:

1. *Independent searches*: no sharing of information occurs among the search programs (only the mapping d_i is active);

2. *Asynchronous cooperative search*: the mapping u_i uses the configurations ψ_j $j \neq i$, and the sharing of information overlaps in a similar way as described by Equations (3) and (4);

3. *Synchronous CA cooperative search*: the mapping u_i uses configurations provided by a cellular automaton;

4. *Synchronous random cooperative search*: the mapping u_i uses configurations generated randomly (according to a normal distribution).

Cooperation schemes 2 and 3 have diffusion dynamics, cooperation schemes 1 and 4 have no diffusion processes. We will compare the trajectory of these different systems in their state space according to the *diversity* of the configurations generated by the transitions. There are different ways to evaluate diversity (Rosca 1995). In this research, we use the Global Hamming Distance between configurations and the Shannon entropy.

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 Shannon entropy E is given by:

$$E = -\sum_{i=1}^{t \times p} p(c_i) \times \log p(c_i) \tag{5}$$

where $p(c_i)$ is the occurrence proportion of the configuration c_i in the $t \times p$ configurations visited ($p(c_i)$ is the number of times the configuration c_i has been generated, divided by the number of configurations generated). If all the configurations generated are different, then the entropy is maximum, while if a single configuration is repeatedly generated, the entropy is at its minimum.

The Global Hamming Distance (GHD) between configurations is calculated in the following way:

- $H(m_i, m_j)$ is the Hamming distance between two configurations m_i and m_j from any of the search paths of the sequential programs involved in the parallel procedure;
- $H^{i+} = \sum_{j=0}^{(t \times p)-1} H(m_i, m_j)$ is the Hamming distance between a configuration m_i and the $(t \times p) - 1$ configurations of the p search paths in the parallel procedure;
- $GHD = \sum_{i=0}^{(t \times p)-1} H^{i+}$, i.e. the Hamming distance between all the configurations of all the search paths of the parallel procedure.

Both approaches to measure diversity can capture variations in the search behavior of the different cooperative procedures (the trajectory of the systems if we view these procedures as dynamical systems). However, since the Shannon entropy measures the repetition of the same configuration, it is more sensitive to variations related to the symbolic evaluation (cost function) of the meta-transitions. The GHD measures the repetition of partial configurations or sub-blocks among the configurations visited. It can read interaction patterns among search programs that will go undetected if we consider interactions only as an exchange of symbolic information, i.e. as solutions of the optimization problem (this is the way the heuristic components of the system see the information they process).

3 Cellular automata theory

Cellular automata have been introduced by Ulam and von Neumann in the 1940s (Burks 1970) to provide a formal framework for the investigation of the behavior of complex, space extended systems. Since then, they have been increasingly used as a tool to study the behavior of dynamical systems (Bonfatti, Gadda and Monari 1994, Langton 1986 and Vichniac 1984). In the present research, the cellular automata theory is used to study the diffusion dynamics of asynchronous cooperative algorithms where neighborhoods overlap. More specifically, CAs are used to simulate the impact that several overlapping exchanges of information have on cooperating programs. The goal of these simulations is to recreate the behavior of cooperating programs with the help of CA controlled communications (interactions). If the simulations are successful, we can use some mathematical properties of CAs to understand how the diffusion dynamics of asynchronous cooperative algorithms affect the search behavior of these procedures.

In this section we give a brief introduction to a particular cellular automata model and then provide some characterizations of CAs as dynamical systems.

3.1 Introduction to one-dimensional cellular automata

Cells of CA could be interconnected according to a one, two or three-dimensional grid, but in the present paper we will be concerned only with one-dimensional CA, i.e. a vector of cells c_i, each connected to its left neighbor cell c_{i-1} and its right neighbor

$$
\begin{array}{lcl}
111 & \rightarrow & 0 \\
110 & \rightarrow & 1 \\
101 & \rightarrow & 0 \\
100 & \rightarrow & 0 \\
011 & \rightarrow & 1 \\
010 & \rightarrow & 1 \\
001 & \rightarrow & 0 \\
000 & \rightarrow & 0
\end{array}
$$

Figure 1: Transition rule 76 of a one-dimensional CA

cell c_{i+1}. We consider a finite-size grid, and we assume spatially periodic boundary conditions are applied, resulting in a circular vector.

Formally, a CA is a triplet (K, Σ, ϕ) where:

- K is the set of states (cells) of the finite automaton;
- Σ is an alphabet;
- ϕ is the transition rule $\Sigma \rightarrow K$ of the finite automaton.

Each finite automaton can be in one of its K possible states. The *neighborhood* $\mathcal{V}$ of a cell c_i is a subset of $2r+1$ adjacent cells having as center the cell c_i. The r cells on either side are the neighbors of cell c_i, as well as c_i itself. The parameter r is referred to as the radius of the neighborhood. The size of the neighborhood is given by $|\mathcal{V}| = N = 2r+1$. Each cell of a cellular automaton has $2r+1$ neighbors. The alphabet Σ corresponds to the set of possible states of the neighborhood $\mathcal{V}$. The cardinality of this set is given by $|\Sigma| = |K|^N$. The state of the neighborhood of a cell c_i, $\mathcal{V}(c_i)$, is the scalar product of the state of the cells in the neighborhood of c_i. The transition rule ϕ is defined by associating a single state of K to one of the possible states of the neighborhood. Since there are $|K|$ possible states for each cell and $|K|^N$ neighborhood states, there is up to $|K|^{|K|^N}$ transitions rules ϕ for a given cellular automaton. For example, assuming that $|K| = 2$ and $r = 1$ (therefore $N = 2r + 1 = 3$), $|\Sigma| = |K|^{2r+1} = 2^3 = 8$ and $|\phi| = |K|^{|K|^{2r+1}} = 2^8 = 256$. A transition rule is specified by giving, for each of the 8 elements $c_{i-1}^{k-1}, c_i^{k-1}, c_{i+1}^{k-1}$ of the alphabet Σ, the value of c_i^k. Figure 1 is an example of such a transition rule. The sequence of 8 bits on the right of the arrows (01001100), once converted in base 10, gives us the number of the transition rule ($01001100 = 76$). We will use these numbers to identify the transition rules in the paper.

3.2 Cellular automata as dynamical systems

The state of each cell of a CA is updated synchronously in discrete time steps, according to a local, identical transition rule ϕ. The input to ϕ is the current state of a surrounding neighborhood of cells (including the current state of the cell itself). The state of all the cells of a cellular automaton at iteration t constitutes the current *configuration* (state) of the CA. Assuming that $|K| = 2$ (each cell has only two states, 0 and 1), and n is the

number of cells, there are 2^n potential configurations, which correspond to the state space of the CA. The dynamics of a cellular automaton is given by:

$$c_i^k = \phi[c_{i-r}^{k-1}, c_{i-r+1}^{k-1}, \ldots, c_i^{k-1}, \ldots, c_{i+r}^{k-1}] \tag{6}$$

where c_i^k is the state of cell c_i at iteration k. The state of a cellular automaton at iteration $k+1$ depends strictly on its current state at iteration k and the transition rule ϕ of the cellular automaton. The dynamics of CA defines the set of configurations visited by the CA as time elapses. This set of configurations is the trajectory of the CA in its state space.

The dynamics of CAs include attractors which can be a single configuration (fixed point), a regular path (simple cycle), a complex series of configurations (non-periodic) or an infinite sequence (called a *strange attractor*). The dynamics of a CA depends on its transition rule. Wolfram has classified the 256 rules of a one-dimensional CA with $r = 1$ according to the type of attractor the CA reaches when transitions follow a given rule. For example, Figure 2 (assuming that white spaces are zeros) represents the dynamics of rule 12 during 8 iterations of a one-dimensional CA (each line represents one iteration). This figure shows that after the second iteration, the CA dynamics stay in the same configuration: the CA has reached a single configuration attractor. Similarly, Figure 3 shows the dynamics of rule 74 for a duration of 11 iterations. Again, after the second iteration, the trajectory of the CA enters an attractor which for this transition rule has a period of 2 iterations (simple cycle).

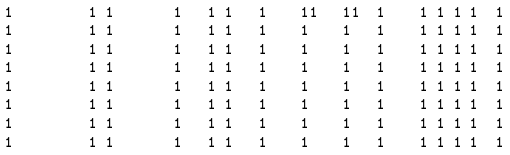

Figure 2: Dynamical behavior of transition rule 12

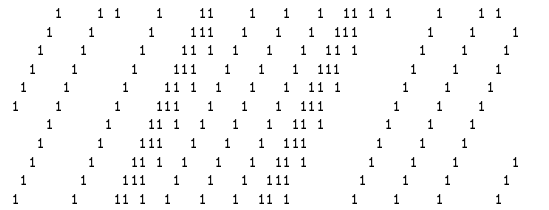

Figure 3: Dynamical behavior of transition rule 74

Dynamics in CAs arises based on the overlap of the neighborhoods of cells. Because of the overlapping of neighborhoods, some rules can induce long chains of correlated dependencies among cells. The number of cells which depend on the state of a cell c_i grows by at most r at each iteration of the CA and on both sides of c_i in the vector. After k iterations, a region of at most $1 + 2rk$ cells could have been affected by the

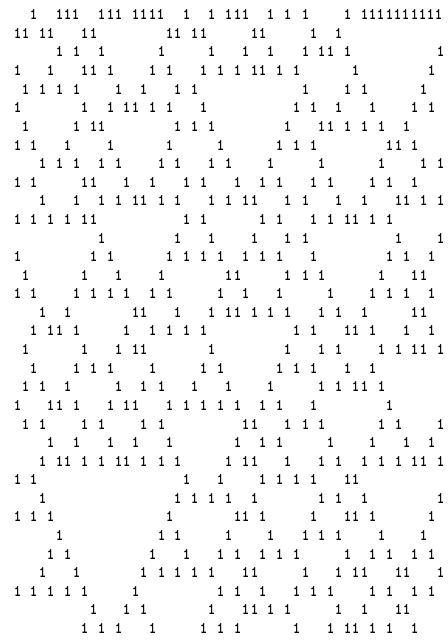

Figure 4: Rule 18 from an initial state generated with p=1/2

state of a cell c_i. We said that cell c_i had an impact on a maximum of $1 + 2rk$ other cells after k iterations of the CA. The *Lyaponov exponent* λ measures the speed of the diffusion process which creates dependencies among cells of a cellular automaton. The one-dimensional CA has only two Lyaponov exponents which measure the speed of the diffusion process on the left and the right of a given cell. Several rules do not have positive Lyaponov exponents (which means that they don't have a diffusion process in both directions of the vector). This is the case for rule 12, where the Lyaponov exponent is null on either side, and for rule 74 where the Lyaponov exponent is null on the right side.

3.3 Choosing CA rules to simulate cooperative procedures

Since we are interested in simulating the cumulative impact of long chains of correlated interactions among cooperating search programs, we have chosen rules where both Lyaponov exponents are positive. For these rules, the state of the cells depends on long chains of dependencies among the cells. For finite one-dimensional CA, these rules generate complex non-periodic dynamics. This is the case for example with the trajectory of rule 18 in Figure 4. The time evolution of rule 18 generates long chains of correlations between values in separated cells. 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. These dependencies translate into a complex structuring process.

Some properties have been identified by Wolfram (1983) for CAs with non-periodic dynamics and the associated complex structuring process taking place in these CAs. Assume ϕ is a transition rule. We deduce a global transition Φ between the set of

configurations:

$$\Phi : \Sigma \rightarrow \Sigma \tag{7}$$

where, as before, Σ is the set of possible configurations of the cellular automaton given a particular transition rule ϕ. Equation 7 simply means that 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 automaton transition rules which generate complex non-periodic attractors, the following hold (Wolfram 1984) :

$$\Omega^{\tau+1} = \Phi\Omega^\tau \subseteq \Omega^\tau \tag{8}$$

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 is the case for rule 18. 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.

Furthermore, it can also be shown that the decrease in the number of configurations in the set Ω can be observed in terms of blocks of sub-configurations which get excluded from the configurations generated by the rule ϕ of the cellular automaton:

Proposition 1 *(Wolfram 1984) A sub-configuration of length l is excluded after τ iterations of a rule ϕ if there is no sub-configuration from the cellular automaton initial configuration of length $(l+2r) \times \tau$ which evolves toward the sub-configuration of length l.*

Proposition 2 *(Wolfram 1984) A sub-configuration of length l is* **newly** *excluded at iteration τ iff no sub-configuration of length $l + 2r$ which exists at iteration $\tau - 1$ can evolve into this sub-configuration, but at least one sub-configuration of length $l + 2r$ newly excluded at iteration $\tau - 1$ would have evolved into this sub-configuration.*

We can observe that blocks of sub-configurations are excluded this way in the trajectory of the cellular automaton in Figure 4. For example, blocks [111] of length 3 are excluded from the initial configuration. The blocks [1101011] of length 7, the blocks [110001011] and [1101001] of length 8, etc. are also excluded.

4 Cooperation schemes

We have implemented three different cooperative procedures based on cooperation schemes inspired from the cellular automata theory and one cooperative procedure which uses randomly generated configurations as cooperation scheme. In this section we detail the implementation of these procedures. We identify the cooperative procedures based on CA by SIM_ϕ where ϕ is the cellular automaton transition rule and those based on randomly generated transitions by SIM_r where r is the seed of the pseudo-random generator.

A cooperative procedure SIM_ϕ is a set of p sequential search programs which exchange information synchronously using a cellular automaton. At every δ iterations,

the sequential programs sent to the cellular automaton the configuration of one solution visited in the previous δ iterations of the program. The configuration send by a program p_i to the cellular automaton is a binary vector m_i of length n (the number of decision variables in the optimization problem). Once the cellular automaton has received the p configurations, it concatenates them into a single binary vector M of length $p \times n$. The position in the vector M of each decision variable $x_k \in m_i$ is given by $(i \times n) + k$. Using this formula, the messages are ordered in the vector M according to the indices of the search programs in the parallel procedure SIM_ϕ.

The vector M constitutes the initial state of the cellular automaton. From this initial state, the cellular automaton performs $\tau = \lceil \frac{2n-1}{2r} \rceil$ iterations using the transition rule ϕ. $\tau = \lceil \frac{2n-1}{2r} \rceil$ makes sure that the search behavior of a program p_i depends on the search behavior of at most two other programs (i.e. each program has two neighbors). Since among the cellular automata used for SIM_ϕ, those having the largest Lyapunov exponents are such that the dependency on a given cell increases by one cell at each iteration, fixing τ to $\lceil \frac{2n-1}{2r} \rceil$ gives us the neighborhood structure we seek. To see that, assume c_a is a cell of the cellular automaton of SIM_ϕ such that c_a corresponds to the decision variable x_a from the message m_i sent by the program p_i to the cellular automaton. Assume c_k is a cell of the cellular automaton such that c_k corresponds to the decision variable $x_k \in m_j$, $j = i-1$ or $j = i+1$. The value of τ corresponds to the number of iterations of the cellular automaton necessary for the state of cell c_a to have an impact on cells c_k, where cells c_k correspond to variables of the configurations sent by programs p_j in the neighborhood of program p_i. For example, let $m_j = \{x_1, x_2, \ldots, x_n\}$ be the configuration returned to program p_j after τ iterations of the cellular automaton under the rule ϕ. The state of each variable $x_k \in m_j$ depends on the state of the cells

$$\{((j \times n) - \tau) + k, \ldots, (j \times n) + k, \ldots, ((j \times n) + \tau) + k\}$$

of the initial vector M. If $\tau = \lceil \frac{2n-1}{2r} \rceil$, the state of $x_0 \in m_j$ is determined by the state of the cells

$$\{(j \times n) - \tau, \ldots, j \times n, \ldots, (j \times n) + \tau\}$$

of M, i.e. the state of the variables sent to the cellular automaton by the search programs p_{j-1} et p_j. If $\tau = \lceil \frac{2n-1}{2r} \rceil$, the value of $x_n \in m_j$ is defined by the cells

$$\{((j+1) \times n) - \tau, \ldots, (j+1) \times n, \ldots, ((j+1) \times n) + \tau\}$$

of M, i.e. the messages sent to the cellular automaton by the search programs p_j et p_{j+1}. Once the cellular automaton has completed its τ iterations, it returns a binary vector to each program p_i which corresponds to the cells $[(i \times n) + 0, (i \times n) + 1, \ldots, (i \times n) + n]$ of the vector M. The state of each variable x_k received by a sequential program is determined by the state of the n cells on each side of c_k in the initial vector M. For rules having a positive Lyapunov exponent, this value of τ implies that the configurations received by a program p_i are correlated to the configurations sent by programs p_{i-1} and p_{i+1} if $\tau = \lceil \frac{2n-1}{2r} \rceil$. Neighborhoods among sequential programs overlap, therefore the repeated interactions with the cellular automata induce a diffusion process among the sequential programs of the cooperative searches (the search behavior of a program p_i at its u-th interaction with the cellular automaton depends indirectly on the behavior of at most $2 \times u$ other search programs). The value of τ is a parameter that can define

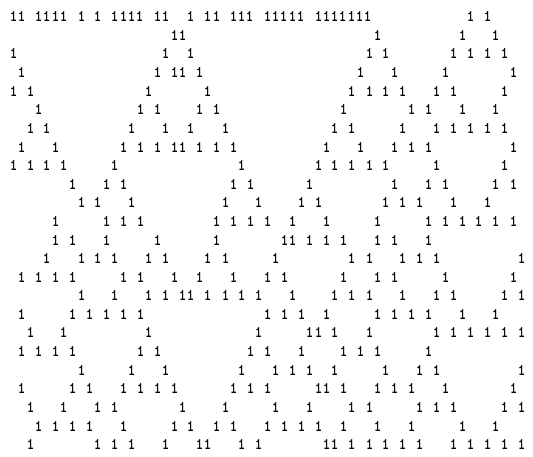

Figure 5: Rule 18 at the first interaction point of procedure SIM_{18}

different sizes of neighborhood structures among the search programs. Changing the size of the neighborhood, for example by making $\tau = 2(\lceil \frac{2n-1}{2r} \rceil)$, will yield a faster diffusion process. We have chosen a neighborhood of two which implies a relatively slow diffusion process in order to simulate as closely as possible the slow diffusion process of asynchronous cooperative searches.

The configurations sent by the cellular automaton are meta-transitions for the sequential programs. The value of τ determines which information is used by the cellular automaton to generate the transitions sent to the sequential programs. Together, the initial value of M, the value τ, and the transition rule ϕ define the set of meta-transitions generated by the cellular automaton.

The cooperative procedures SIM_r are also synchronous. At every δ iterations, a program generates random binary vectors of length n and returns a different vector to each of the sequential search programs. Once the sequential programs have received a configuration from the cellular automaton or the random number generator, they restart their exploration of the solution space from the configuration vector they have received (which may not be a feasible solution).

We have implemented and tested 4 different cooperative procedures based on these cooperation schemes: SIM_{12}, SIM_{18}, SIM_{22} (which are based on cellular automata, respectively, implementing the transition rules 12, 18 and 22) and SIM_R. Rules 18 and 22 have been chosen because of the type of attractors they induce in CAs and their diffusion processes (as explained in the previous section), while rule 12 has no diffusion dynamics and has been chosen for comparison purposes.

The configurations randomly generated and those obtained from CAs, interfere with the sequential control structure of the search programs. Therefore they have an impact on the search behavior of those programs (this is no different than the exchange of information from previously visited regions of the solution space). For example, by comparing Figures 5 and 6, we can have a visual appreciation of what is accomplished by the cellular automata with rule 18 on the search behavior of the sequential programs. The initial vector M in Figure 5 is based on configurations of solutions sent to the

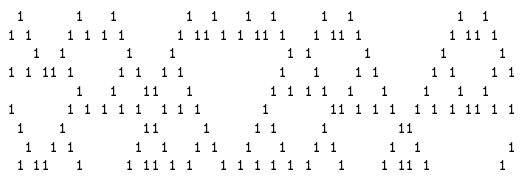

Figure 6: Rule 18 at the second interaction point of procedure SIM_{18}

cellular automaton after 25 iterations of the sequential search programs (it is the first interaction of the sequential programs with the cellular automaton). The τ iterations of the cellular automaton transform the initial configuration M into a configuration from where obviously some sub-configurations are excluded. Figure 6 shows the initial vector M after 50 iterations of the sequential programs (the second interaction with the cellular automaton). In Figure 6, the vector M is structurally very different from the vector M in Figure 5, for example the sub-configurations [1111], [11111] and [1111111] are not present in the vector. This is a good indication that the transitions provided by the cellular automaton have an impact on the search behavior of the sequential programs between the iterations 25 to 50.

5 Experimentations

Tests have been executed for cooperative algorithms using 6 different cooperation schemes: SIM_{12}, SIM_{18}, SIM_{22}, SIM_R, IS (Independent Searches) and ACS (Asynchronous Cooperative Search). "IS" refers to independent searches; this cooperation scheme does not involve any exchange of information. "ACS" refers to asynchronous cooperative search. For ACS procedures, the cooperation scheme consists to take one configuration from a search program p_i and use it as the new current configuration of another search program p_j. The choice of the configurations is based on the cost function. This exchange takes place at a particular point in time where the two programs are identified as neighbors. Programs become part of a neighborhood based uniquely on conditions related to their internal state. Neighborhoods overlap, which means that a program p_i can be part of a neighborhood Y at one point in time and then according to its internal state can become part of another neighborhood Z later on during the parallel computation. This overlapping of neighborhoods gives raise to the diffusion of information shared with program p_i in neighborhood Y to one or several other programs p_j in the neighborhood Z.

Each cooperative procedure is tested on 12 different instances of the location-allocation problem introduced in Crainic, Gendreau, Soriano and Toulouse (1993). 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 the paper above. Each program executes 300 TS iterations. Tests are conducted on a network of Sparc workstations. The location-allocation problem in this paper has two types of decision variables: design variables, which are boolean (0/1); and continuous flow variables. The tabu search method runs in the space defined by the design variables,

therefore the solution space is given by $X^n \subseteq B^n$.

The diversity of the different cooperative procedures is shown in the Table 1 (Shannon entropy) and Table 2 (GHD: Global Hamming Distance). In these two tables, the lines "Avrg" represent the average diversity of the 12 problem instances.

5.1 The Shannon entropy

The Shannon entropy is not affected by the cooperation scheme, except for ACS and SIM_{12} where it is slightly lower. In the case of ACS procedures, transferring a configuration from one programs to another implies that the configuration appears in both programs, which creates a slight reduction of diversity for this scheme. This is also the case for the SIM_{12} scheme. The Lyapunov exponents of rule 12 are null. As shown in Figure 2, the cellular automaton with rule 12 reaches a single point attractor after its first iteration. Then the same configuration is repeated. Consequently, the configuration of the initial vector M is almost not affected by the transitions of the cellular automaton. The cellular automaton returns identical or very similar configurations to the search programs. When that same configuration is returned, this configuration is considered to have been visited at least twice, which has an impact on the Shannon entropy.

5.2 The global Hamming distance

We observe from Table 2 that the cooperation schemes which do not have diffusion dynamics have substantially higher GHD values compared to the cooperation schemes involving diffusion dynamics. It indicates that diffusion processes are affecting the diversity of ACS procedures. Since the GHD values of SIM_{18} and SIM_{22} are relatively similar to GHD values of ACS, we can be confident that the simulation by the cellular automata captures important aspects of the cumulative impact of the diffusion dynamics on the search behavior of ACS procedures. We now give detailed explanations about the diversity of the cooperation schemes involving diffusion dynamics.

5.2.1 GHD values of SIM_{18} and SIM_{22}

The structuring process taking place in the configurations of a CA such as the one depicted in Figure 4, corresponds according to Equation (8), 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 Propositions (1) and (2) into an exclusion of some sub-configurations (blocks of length l) from the states reached by the CA. (For example, in the last configuration of Figure 4, 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 4. 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

Prob.	IS	ACS	SIM_{18}	SIM_{22}	SIM_{12}	SIM_R
	4 sequential programs					
P1	9.3912	9.3082	9.5646	9.3536	8.4967	9.5245
P2	9.6434	9.3995	9.7539	9.7232	9.5737	9.6487
P3	9.6346	9.6031	9.6279	9.5873	9.3288	9.6729
P4	9.7647	9.6047	9.7907	9.6392	9.5997	9.7055
P5	9.6282	9.3968	9.6359	9.6558	8.9991	9.66102
P6	9.6508	9.3898	9.5997	9.7512	9.5897	9.7536
P7	9.2656	9.3167	9.0945	9.2185	7.9872	9.2541
P8	9.4081	9.3708	9.4381	9.4381	8.7966	9.6416
P9	9.6961	9.0218	9.5289	9.5962	9.1441	9.6226
P10	9.6327	9.4604	9.5787	9.5494	9.3027	9.6743
P11	9.1469	9.2975	9.3635	9.6155	9.2178	9.6076
P12	9.7223	9.3240	9.6889	9.6122	9.4783	9.7195
Avrg	9.5487	9.3744	9.5554	9.5617	9.1262	9.6238
	8 sequential programs					
P1	10.4336	10.1299	10.2654	10.5063	9.4018	10.3329
P2	10.7333	10.5222	10.7601	10.7638	10.6519	10.7308
P3	10.7247	10.5751	10.6297	10.6630	10.3777	10.7241
P4	10.8408	10.6167	10.7936	10.8069	10.6858	10.8333
P5	10.7134	10.4826	10.6989	10.7127	10.2533	10.7489
P6	10.7363	10.5126	10.7684	10.7818	10.6581	10.8324
P7	10.2222	10.1173	9.9785	10.0702	8.8972	10.1997
P8	10.5694	10.4113	10.5399	10.5475	9.7734	10.5402
P9	10.6569	10.5751	10.5276	10.5465	10.0988	10.6685
P10	10.7233	10.5219	10.5717	10.6766	10.4524	10.7472
P11	10.4317	10.3511	10.5596	10.5925	10.1297	10.6109
P12	10.7479	10.3811	10.6685	10.7083	10.3498	10.7400
Avrg	10.6279	10.4331	10.6685	10.6147	10.1442	10.6424
	16 sequential programs					
P1	11.3045	10.9599	11.2980	11.3321	9.7500	11.2085
P2	11.7505	11.4765	11.7881	11.7673	11.4833	11.7594
P3	11.6529	11.3618	11.6307	11.6571	11.2887	11.7265
P4	11.6694	11.5792	11.7975	11.7844	11.5974	11.7767
P5	11.6677	11.5217	11.7149	11.7338	11.3117	11.7380
P6	11.7344	11.5159	11.7446	11.7564	11.5942	11.7690
P7	11.1474	10.9154	10.8722	10.9740	9.3263	10.9593
P8	11.5064	11.2043	11.5272	11.6252	10.5002	11.6160
P9	11.6259	10.7628	11.4207	11.5463	10.7860	11.6062
P10	11.7002	11.4778	11.6024	11.6583	11.3487	11.7591
P11	11.5168	11.2194	11.5381	11.5127	11.0427	11.6080
P12	11.7765	11.3283	11.6856	11.7162	10.9976	11.7912
Avrg	11.5877	11.2769	11.5517	11.5887	10.9189	11.6098

Table 1: Shannon Entropy

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 how dynamics of the complex non-periodic attractors of rules 18 and 22 reduce the diversity of the parallel procedures SIM_{18} and SIM_{22}.

5.2.2 GHD values of asynchronous cooperative searches

The sharing of information in ACS procedures interferes whith the control structure of the search programs through the transfer of a configuration m_i between two neighbor programs p_i and p_j. Once m_i has been transferred 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 become a neighbor of a program p_k to which it will send a configuration m_j. The meta-transitions based on configurations m_i and m_j are said to

Prob.	IS	ACS	SIM_{18}	SIM_{22}	SIM_{12}	SIM_R
	4 sequential programs					
P1	10472042	6544523	6441348	6732640	4398172	7457976
P2	12165930	10196267	9895244	10236394	7154056	1154810
P3	10128906	8169040	7431169	7691982	6209248	8509727
P4	12231950	10511216	11278506	11494450	8641152	11746172
P5	11609382	9779934	9628766	11068821	6344251	11076438
P6	12103916	9953606	10560432	11299182	7804042	11637816
P7	8177905	7060246	4291314	5490248	3307584	5798190
P8	9198614	7135749	5885788	6518142	3962889	7347155
P9	8724796	6218284	5854062	7084810	4448600	7625296
P10	10810398	8414525	6900029	7936606	5870548	8295301
P11	8826493	6842760	6630424	7178887	5737208	7901108
P12	9507103	7700852	7248392	8212247	6354552	8628270
Avrg	10329786	8210583	7670456	8412034	5852691	8964296
	8 sequential programs					
P1	37283682	27127187	24140619	26319684	17793228	29120668
P2	47464746	40753078	39932837	41740142	30064208	43658906
P3	38167472	31303376	31460412	33270520	25050773	33215412
P4	47973642	41710930	46260170	45816860	31612130	46961650
P5	46324714	39751248	42341262	43257914	30369833	44605104
P6	47776650	40607090	41580526	43382848	33592608	46365316
P7	29565440	27291980	16391230	18408216	12799844	23569269
P8	34359038	29715919	2412901	25140624	17424814	29260554
P9	33010204	31303376	22821213	23971320	18311246	28595215
P10	39418660	34295182	28228062	30121012	24838178	34225418
P11	35366122	28897783	26078993	27226834	21232303	31109265
P12	37560875	31052398	30649343	31881154	25287575	35623672
Avrg	39522603	33650795	31167806	32544760	24031395	35525870
	16 sequential programs					
P1	148920018	110753063	104493843	113240100	72743208	126046706
P2	191306992	163554218	166303252	168844422	120956920	181473838
P3	155312802	125940704	128374782	134041967	104136450	139155130
P4	196044322	165787158	183742609	191065379	128352932	189290436
P5	191107700	162373970	170591888	181106310	138984060	184009239
P6	193173546	168933312	16830013	182610343	136885969	189216020
P7	120542518	113449220	73846493	80754274	54863070	96098550
P8	141001659	121150512	104235880	113217162	78589764	125217538
P9	139580696	109565144	96767210	107074222	81437266	117778878
P10	157462065	140055168	120289894	123153253	107360308	143126157
P11	145271065	139132459	112898006	117279048	89240238	133004154
P12	155515648	126423019	125331585	133597682	100376740	145774148
Avrg	161269919	137259828	129597964	137165346	101160577	147515899

Table 2: Global Hamming Distances

be correlated to each other and are part of the same diffusion process. Figure 7 shows a correlated sequence of meta-transitions (diffusion process) which interfere with the control structure of several sequential programs. Configuration m_0 is sent to program p_1, this program use m_0 as an initial solution to which several consecutive search method transitions are applied, which changes the state of some variables (those with value "2" in the last configuration of program p_1 in Figure 7). When configuration m_1 is transferred 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 transfer of m_1 from p_1 to p_2 is equivalent to copying the state of several variables from a configuration of program p_0 to program p_2. The state of variables in a same diffusion process which are not changed by the sequential control of the programs are copied across several search programs. For example, in Figure 7 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

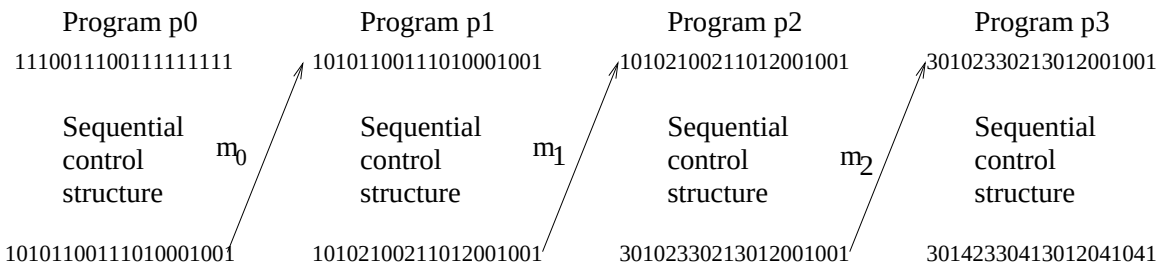

Figure 7: Diffusion process among search programs

(e.g. 010) in the configurations of several programs is the main factor contributing to the reduction of the GHD values in the ACS procedures.

5.3 Shannon entropy and GHD

From Table 1, we observe that Shannon entropy values are very stable, which indicate that the trajectory of the cooperative procedures (in terms of the configurations) is not statistically affected by the cooperation scheme or the selection mode of the meta-transitions. In terms of the diversity of sub-configurations (GHD), there are significant statistical differences between the trajectories of the systems depending if the cooperation scheme involves diffusion dynamics or not. Discrepancies between Shannon entropy and GHD values occur mainly in cooperation schemes with diffusion dynamics (ACS, SIM_{18} and SIM_{22}). The GHD values are lower for those cooperation schemes and they are proportionally lower than Shannon entropy values for the ACS scheme (this can be measured by taking the ratio ACS/IS for both the GHD and Shannon entropy values). Diffusion processes among cooperating programs seem to be what cause the GHD values to be lower for ACS, SIM_{18} and SIM_{22} procedures.

The lower GHD values for ACS procedures are obtained by copying the same sub-configurations across several search programs. For the procedures SIM_{18} and SIM_{22}, the same effect is obtained by injecting from a CA similar configurations (identical sub-configurations) in several search programs. The GHD values of SIM_{18} and SIM_{22} procedures are similar to ACS procedures although the meta-transitions are not evaluated. In fact, this is not a surprise since sub-configurations rather than full configurations are propagated in diffusion processes. The selection mode may affect which configuration is copied from a program p_i to p_j, but it has no impact statistically on the sub-configurations which are propagated in diffusion processes. In fact, if we assume that the best configurations (in terms of the cost function) are distributed evenly among the configurations generated by the search method transitions of a program, then it does not matter for the diffusion processes if the configurations to be copied from one program to another are selected randomly or according to the cost function. Statistically, similar sub-configurations (in terms of size and quantity) are going to be copied across several search programs. Therefore, there are no links between the mode of selection of the meta-transitions, the diffusion dynamics, and the statistical properties of the trajectory of ACS procedures (considered as dynamical systems). It

is clearly a mistake in the current design of cooperative algorithms to assume that the cost function can control the diffusion dynamics. This assumption is not substantiate by our results.

Entrenched behind the assumption that cost function can take care of diffusion dynamics, is the believe that diffusion dynamics are propagating symbolic information. Again our results seem to prove this wrong. Symbolic information are the data from the symbolic part of cooperative procedures (the search methods) which are used by the optimization strategy of the search method (configurations of the solution space are a significant part of these data). Although configurations are copied from one program to another, they are not propagated in the diffusion processes. On the other side, sub-configurations, which are propagated, do not constitute symbolic entities that can be used by the optimization strategy of the search methods. Although they are not used by the search methods, sub-configurations are copied by the meta-transitions to several programs. Through this way (which is how shared information interfere with the sequential control structure of the search programs) they have a strong impact on the search behavior of cooperating programs, which shows-up in the GHD values of the ACS procedures. Given that diffusion dynamics propagate chunks of data that cannot be used by the search methods, we are not inclined to believe that they helping cooperating programs in their optimization strategy or that they give an edge to ACS procedures over independent searches.

In a broad sense, the present study shows that diffusion processes exist in cooperative algorithms, they affect the search behavior of cooperating programs, but their impact on the optimization strategies seems to be uncontrolled. It will certainly help cooperative algorithms if the diffusion processes propagated information that can be processed by the search methods. There are already promising developments in that direction. One possible approach to achieve this goal consists to design new cooperative algorithms which focus on the natural dynamics observed in the simulations or studies of existing cooperative algorithms. One should try to simplify how the cumulative impact of shared information can build-up among cooperating programs. Such simplifications can be achieved by restricting the cooperation schemes in terms of the neighborhood structure allowed among cooperating programs, the state space that can be explored by each search program (the search space), and the relations that exist among the search space and the information shared with neighbor programs. Simplifications of cooperation schemes may yield dynamics for which analytical models can be derived. We are designing such simplified cooperative algorithms. An application to the graph partitioning problem has been considerably more successful on controlling the cumulative impact of diffusion processes on the search behavior of cooperating programs (see Toulouse, Thulasiraman and Glover 1998).

6 Conclusion

For cooperative algorithms, the sharing of information among the search programs creates an inextricable network of dependencies which affects in a very complex manner the search behavior of those programs. The design of most cooperative algorithms does not address directly this issue. Rather it is assumed that by selecting information for

sharing according to the cost function of the optimization problem, cooperation, not only will not mislead the optimization strategies, but will help them to find better solutions.

In this paper we model the sharing of information and the dependencies it create as diffusion processes, and their impact on the search behavior of cooperating programs as diffusion dynamics that affect and interact with the internal dynamics of sub-systems from a space-extended dynamical system. Since it is not possible to develop analytical models to see how the diffusion dynamics affect the search behavior of cooperating programs, we use statistical measures such as the Shannon entropy and the Hamming distance to evaluate this impact. We compare the global Hamming distance and the Shannon entropy of the configurations visited by asynchronous cooperative procedures with other search programs which are not sharing information, use information provided randomly or by cellular automata which have converging dynamics (evolve toward lower entropy).

Our findings indicate that diffusion processes of asynchronous cooperative procedures propagate the same type of information as programs which cooperate based on data provided by cellular automata, i.e. meaningless (from an optimization point of view) sub-configurations. The cumulative impact of the interaction among search programs (the diffusion dynamics) affects the trajectory of cooperative procedures (understood as dynamical systems). However the new trajectory (compared with search programs which do not share information) may not overlap with better regions of the solution space, and consequently cooperation may not yield better performance than independent searches.

There are two important (and related) observations that come out of this research and which can be helpful in improving the design of future cooperative algorithms. One is the realization that the dynamics set off by the sharing of information are not well controlled in terms of their effect on the exploration of the solution space by cooperating programs. It would help if we find a way to subordinate these dynamics to the optimization goals of the search programs. Second, cooperative procedures are parallel processing dynamical systems; we should come-up with control structures which reflects this reality. For example, it is clear that the sub-configurations play naturally a key role in controlling the asynchronous cooperative procedures (although they are not an "optimization kind" of control structure). It might be profitable to understand known control mechanisms of dynamical systems theory (such as diffusion dynamics, feedback loops, etc.) and find ways to adapt them to produce search behaviors favorable to the optimization logic. Then, with their combination of the symbolic model of computation (the search heuristics) and the connectionnist model of computation (interactions among the search programs), cooperative algorithms can constitute a powerful approach for combinatorial optimization problems and other types of computationally complex problems.

References

[1] F. Bonfatti, G. Gadda, and P.D. Monari. Simulation of Dynamic Phenomena by Cellular Automata. *Comput. & Graphics*, 18(6):831–836, 1994.

[2] A.W. Burks. *Essays on Cellular Automata*. University of Illinois Press, Urbana, 1970.

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

[4] 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.

[5] T.G. Crainic, M. Toulouse, and M. Gendreau. Parallel Asynchronous Tabu Search for Multicommodity Location-Allocation with Balancing Requirements. *Annals of Operations Research*, 63:277–299, 1995.

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

[7] C.G. Langton. Studying Artificial Life with Cellular Automata. *Physica D*, 22:120–149, 1986.

[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, 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.

[12] J.P Rosca. Entropy-Based Adaptative Representation. Report TR-NRL02, University of Rochester, Rochester NY, 1995.

[13] M. Toulouse, C. Crainic, and B. Sansó. Systemic Behavior of Cooperative Search Algorithms. *Submitted to Parallel Computing*, 1998.

[14] M. Toulouse, F. Glover, and K. Thulasiraman. A Multi-Scale Cooperative Search with an Application to Graph Partitioning. Publication, School of Computer Science, University of Oklahoma, Norman, OK, 1998.

[15] G.Y. Vichniac. Simulating Physics with Cellular Automata. *Physica D*, 10:96–116, 1984.

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