Catalytic Search in Dynamic Environments

Lidia Yamamoto[1] and Wolfgang Banzhaf[2]

[1]Computer Science Department, University of Basel, Switzerland
[2]Memorial University of Newfoundland, St. John's, NL, Canada
Lidia.Yamamoto@unibas.ch, banzhaf@cs.mun.ca

Abstract

Catalytic Search is an optimization algorithm inspired by random catalytic reaction networks and their pre-evolutionary dynamics. It runs within an Artificial Chemistry in which reactions can be reversible, and replication is not taken for granted. In previous work one of us had shown that although inherently slower than Evolutionary Algorithms, Catalytic Search is able to solve simple problems while naturally maintaining diversity in the population. This is a useful property when the environment may change.

In this paper, we compare the performance of Catalytic Search and a Genetic Algorithm in a dynamic environment represented by a periodically changing objective function. We investigate the impact of parameters such as temperature, inflow/outflow rate, and amount of enzymes. We show that Catalytic Search is generally more stable in the face of changes, although still slower in achieving the absolute best fitness. Our results also offer some indications on how catalytic search could either degenerate into random search, or progress towards evolutionary search, although the latter transition has not been fully demonstrated yet.

Introduction

Artificial chemistries have been used to understand the origin of evolution from a pre-evolutionary, random initial state (Fontana and Buss (1994); Dittrich and Banzhaf (1998)), to devise bottom-up chemical computing algorithms for emergent computation (Banzhaf et al. (1996); Dittrich (2005)), and to build new optimization algorithms (Banzhaf (1990); Kanada (1995); Weeks and Stepney (2005)), among other usages. The motivation for the present work lies at the intersection of these three application domains. We are interested in exploring the emergent computation properties of artificial chemistries for the construction of beamed search schemes able to optimize solutions to user-defined problems. Instead of a top-down, pre-designed optimization algorithm, optimization could be regarded as a computation task to emerge from the bottom up, as an outcome of molecule interactions. In this context, it is worth determining the conditions for the emergence of optimization, of which evolution is only one example.

Bagley and Farmer (1991) showed that primitive metabolisms called *autocatalytic metabolisms* can emerge in an artificial chemistry where polymers undergo reversible polymerization reactions. One of the conditions for the emergence of such metabolisms is to drive the system out of equilibrium by a constant inflow of molecules from the food set, accompanied by a non-selective dilution flow. In this case, some reactions may be boosted by *catalytic focusing*: starting from a random soup of molecules, the system ends up focusing most of its activity and mass into a few types of molecules, self-organizing into autocatalytic reaction networks that consume food molecules to produce longer polymers. The molecules taking part in this autocatalytic core can be regarded as primitive metabolisms.

In previous work, Yamamoto (2010) proposed *catalytic search*, an optimization scheme inspired by catalytic focusing. Catalytic search is based on a pre-evolutionary chemistry (Nowak and Ohtsuki (2008)), where reactions might be reversible, and replication is not taken for granted. The reaction energy functions are assigned such that reactions towards fitter products are favored. The selective pressure in catalytic search comes from the differences in reaction rates for different molecules in the reactor. These differences can be amplified selectively by enzymes: some reactions can be accelerated by enzymes that decrease the activation energy barrier necessary for them to occur. Due to the absence of direct replication, he performance of such scheme lies between that of a random search, and that of an evolutionary algorithm. In spite of such apparent weakness, catalytic search and related chemical schemes have many interesting properties, as pointed out by Weeks and Stepney (2005): the potential to undo wrong computations or to decompose bad solutions through reversible reactions; the ability to steer the reaction flow towards the production of good products by shifting the equilibrium distribution of molecules; a certain robustness to noisy fitness feedback; and the prevention of premature convergence through a natural tendency to generate and maintain diversity in the population. This paper focuses on the latter property.

Catalytic Search

In this section we summarize the catalytic search algorithm by Yamamoto (2010), and introduce our own modifications: an improvement of the original enzyme matching scheme, and its adaptation to run continuously in dynamic environments.

Catalytic search works as follows: initially, a random soup of monomers (letters from an alphabet Σ) is generated. These monomers later concatenate into polymers (strings of symbols from Σ). Each polymer represents a candidate solution to the problem to be solved. At every time step, two molecules (monomers or polymers) are chosen for collision. They react with a probability k, which is also the *kinetic coefficient* of the reaction. If they react, a *crossover* of the two molecules is produced, and the two resulting molecules are injected into the soup. The educts are consumed in the process. The collision is elastic with probability $(1-k)$.

A crossover reaction can be written as follows:

$$A + B \underset{k_r}{\overset{k_f}{\rightleftharpoons}} C + D \tag{1}$$

where A, B, C and D are strings from an alphabet Σ, k_f is the coefficient of the forward reaction, and k_r is the coefficient of the reverse reaction. An example for strings from $\Sigma = \{a, b, c, d\}$ is:

$$abdba + ccbdd \underset{k_r}{\overset{k_f}{\rightleftharpoons}} abdbdd + ccba \tag{2}$$

Crossover is a mass-conserving operation, i.e. it conserves the total number of symbols before and after the reaction. Concatenation occurs as a special case of crossover where the crossover points are the beginning and end of each string, respectively.

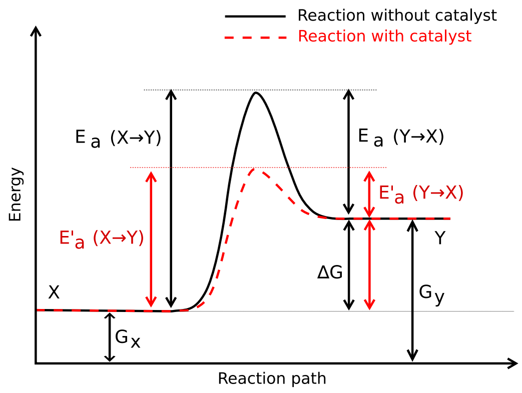

Figure 1: Potential energy changes during catalysed and uncatalyzed chemical reactions. From Yamamoto (2010).

Once the molecules have collided, the reaction only occurs if the molecules have sufficient kinetic energy in order to overcome the *activation energy barrier* (E_a) needed for the reaction. A *catalyst* is a substance that participates in a chemical reaction by accelerating it without being consumed in the process. Its effect is to lower the reaction's activation energy peak, thereby accelerating the reaction, while leaving the initial and final states unchanged. This acceleration comes from the fact that the coefficient k decreases exponentially with the activation energy, following the *Arrhenius equation* from chemistry:

$$k = Ae^{-\frac{E_a}{RT}} \tag{3}$$

where A is the so-called *pre-exponential factor* of the reaction, E_a is its *activation energy*, T is the absolute temperature, and R is a constant.

Figure 1 shows the energy diagram for a typical reversible reaction, where the effect of catalysis is highlighted with a red dotted line. The difference in potential energy before and after the reaction is given by ΔG:

$$\Delta G = G_p - G_e \tag{4}$$

where G_e is the potential energy of the educts, and G_p that of the products. In Figure 1, $G_e = G_X$, $G_p = G_Y$, and $\Delta G > 0$ if the reaction moves from left to right (i.e. in the direction from X to Y, the forward reaction); in the direction of the reverse reaction (from Y to X), we have $G_e = G_Y$, $G_p = G_X$, and $\Delta G < 0$. In this figure, the reverse direction is favored since it leads to more stable products (i.e. $\Delta G < 0$), while the forward direction is unfavored ($\Delta G > 0$). The reverse direction sees a lower activation energy than the forward direction ($E_a(Y \rightarrow X) < E_a(X \rightarrow Y)$) therefore it will be faster on average. Catalysis further reduces this barrier, accelerating the reaction in both directions ($E'_a(Y \rightarrow X) < E_a(Y \rightarrow X)$ and $E'_a(X \rightarrow Y) < E_a(X \rightarrow Y)$).

In order to steer the system towards the production of fitter solutions, in catalytic search the potential energy of a molecule is mapped to the fitness of the solution that it represents. The fitness function must be designed such that lower values indicate better fitness, for instance, a shorter distance to the optimum, or a lower cost of the solution. The educt and product energies are calculated as the sum of the fitness of the molecules involved:

$$G_e = f(A) + f(B) \tag{5}$$
$$G_p = f(C) + f(D) \tag{6}$$

where $f(i)$ is the fitness of solution i. In this way, fitter solutions have a lower potential energy and are therefore more stable. The production of fitter solutions (i.e. with lower potential energy) is favored ($\Delta G < 0$), whereas the production of poorer solutions is unfavored ($\Delta G > 0$), which is the desired effect.

The activation energy for a reaction is further mapped to the estimated computation cost of producing a solution: solutions that are more difficult to compute must overcome a higher energy barrier, and therefore will be less likely to occur. This leads to a form of double-objective optimization scheme that seeks to optimize the fitness of the solution as well as the efficiency of its computation; these two objectives can be balanced via a proper choice of energy functions.

An increase in activation energy ΔE_a corresponding to the cost of the operation is then added on top of the highest potential energy G. ΔE_a corresponds to the portion $E_a(Y \to X)$ in Figure 1.

The activation energies of the forward and reverse reactions, E_{af} and E_{ar} respectively, are:

$$\text{if} \quad \Delta G \leq 0 \quad \begin{cases} E_{af} = \Delta E_a \\ E_{ar} = \Delta E_a - \Delta G \end{cases} \tag{7}$$

$$\text{if} \quad \Delta G > 0 \quad \begin{cases} E_{af} = \Delta E_a + \Delta G \\ E_{ar} = \Delta E_a \end{cases} \tag{8}$$

The coefficients k_f and k_r follow a simplified form of the Arrhenius equation:

$$k_f = e^{-\alpha E_{af}/T} \tag{9}$$

$$k_r = e^{-\alpha E_{ar}/T} \tag{10}$$

where α is a configuration parameter of the algorithm (currently set to $\alpha = 1$), and T is the temperature of the reactor.

This scheme is able to steer the flow of production of candidate solutions towards better ones, without explicit replication, and without an explicit memory of which molecules produced good solutions. The search process is guided by the differences in reaction rates to move from one pair of candidate solutions to another.

Enzymes

The energy-based reaction steering scheme described above is further complemented with an enzymatic step: reactions may be catalysed by enzymes that decrease the needed activation energy. In nature, enzymes act on both forward and reverse sides of the reaction, therefore the equilibrium concentrations do not change. In contrast, the enzymes used in catalytic search only facilitate the forward reaction in the direction of fitter products.

Enzymes are kept in a separate pool. When two molecules collide, if the reaction results in $\Delta G < 0$, i.e. in better fit products, then an enzyme might be created for this reaction, with a probability p_c proportional to the amount of improvement $|\Delta G|$. The next time similar molecules collide, the enzyme can be used to facilitate their reaction, by lowering the corresponding ΔE_a.

In the original catalytic search scheme only exact match between enzyme and substrates was supported. In this paper, we extend the matching scheme such that enzymes bind to their substrates with a certain affinity, proportional to how well their strings match. With this scheme, an enzyme may accelerate similar reactions, and a reaction may benefit from the combined catalytic effect of similar enzymes. For this purpose, we have modified the format of the enzymes in the original catalytic search scheme in order to take into account the strength of matching between enzyme and substrates. In our scheme, enzymes are built from chemical reactions as follows. A generic crossover reaction between two educt strings s_1 and s_2 can be written as:

$$s_{1a}s_{1b} + s_{2a}s_{2b} \to s_{1a}s_{2b} + s_{2a}s_{1b} \tag{11}$$

where s_{ij} are the substrings in s_i separated by the chosen crossover points. An enzyme for this reaction is a string of the form: "$s_{1a}|s_{1b}|s_{2a}|s_{2b}$", with the vertical bar "|" indicating the crossover points. The enzyme uniquely identifies the reaction, and can therefore be used to represent it in molecular form, constituting a memory of past successful reactions.

We use the similarity between the enzyme and the concatenated substrates as the affinity metric. The similarity is the number of matching positions in the alignment between the two strings. For the example of Reaction (2), the corresponding perfectly matching enzyme is "$abd|ba|cc|bdd$". If another reaction between similar strings with similar crossover point happens, say, one described by enzyme "$abb|a|cc|bd$", then the similarity between the two corresponding enzymes is high (10 over a maximum of 11 in this example), leading to a higher catalytic enhancement. The similarity is further normalized by the length of the smallest of the two strings, such that shorter polymers also have a chance to get catalysis. More exactly, the binding strength function between two strings s_1 and s_2 is defined as $bind(s_1, s_2) = similarity(s_1, s_2)/min(length(s_1), length(s_2))$.

Once two molecules collide and their crossover points are decided, a small number of enzymes (subset B) are drawn at random from the enzyme pool, and their matching strengths are calculated with respect to the perfect enzyme c for the reaction. The contributions of all enzymes are added up together: $s_c = \sum_{b \in B} bind(b, c)$. The sum of the strengths is then used to calculate the reduction in activation energy contributed by the enzymes. If $s_c \geq 1$, the new activation energy becomes:

$$\Delta E'_a = \frac{\Delta E_a}{s_c} \tag{12}$$

else ΔE_a remains unchanged.

In order to make sure that the enzyme pool is periodically refreshed and does not grow unbounded, enzymes are subject to a non-selective dilution flow beyond the maximum capacity of the enzyme pool, C_{max}.

We have further modified the algorithm to run continuously, not stopping when a solution is found, in order to run it in dynamic environments. The updated algorithm is shown in Algorithm 1.

Algorithm 1 Catalytic Search Algorithm

```
1: $S$: multiset of candidate solutions
2: $C$: pool of enzymes (catalysts)
3: $C_{max}$: maximum capacity of $C$
4: **initialization:**
5: $S =$ random soup of $N$ monomers $m \in \Sigma$
6: $C = \emptyset$
7: **while** $true$ **do**
8:     expel two random molecules $e_1$ and $e_2$ out of $S$
9:     $(i_1, i_2) =$ random crossover points within $e_1$ and $e_2$
10:    $(p_1, p_2) \leftarrow$ crossover$(e_1, e_2, i_1, i_2)$
11:    $G_e =$ fitness$(e_1)+$ fitness$(e_2)$
12:    $G_p =$ fitness$(p_1)+$ fitness$(p_2)$
13:    $\Delta G = G_p - G_e$
14:    $E_a = (|e_1| + |e_2|)/2$
15:    **if** $\Delta G > 0$ **then**
16:        $E_a \leftarrow E_a + \Delta G$
17:    **else if** $\Delta G < 0$ **then**
18:        $c =$ enzyme$(e_1, e_2, i_1, i_2)$
19:        $B \leftarrow$ draw $n_c$ enzymes from $C$
20:        $s_c = \sum_{b \in B}$ bind$(b, c)$
21:        **if** $s_c \geq 1$ **then**
22:            $E_a \leftarrow E_a / s_c$
23:        **end if**
24:        $p_c = |\Delta G| / G_e$
25:        add another instance of $c$ to $C$ with probability $p_c$
26:        **while** $|C| > C_{max}$ **do**
27:            destroy a random catalyst from $C$
28:        **end while**
29:    **end if**
30:    $k_f = e^{-\alpha E_a / T}$
31:    **if** dice$(k_f)$ **then**
32:        inject new products $p_1$ and $p_2$ into $S$
33:    **else**
34:        inject educts $e_1$ and $e_2$ back to $S$
35:    **end if**
36: **end while**
```

Catalytic search steers the flow of chemical reactions by acting primarily on the rate coefficients rather than on the concentrations. Therefore it has a natural tendency to keep a diversity of molecules in the reactor, some of which are rarely used because of a slow reaction speed, but nevertheless stay present at some concentration. These molecules could become useful in the future, for instance when the environment changes. This provides a simple way to keep a pool of alternative solutions in the population, and to switch to different solutions by preferentially choosing different reaction pathways to construct alternative solutions using the elements in the pool. In this paper we perform experiments to support this claim.

Genetic Algorithm in a Chemistry

For comparison purposes, a Genetic Algorithm (GA) is implemented within a similar artificial chemistry. This GA was briefly introduced in (Yamamoto (2010)). Here we describe it in more detail for completeness. It is a variation of a Steady-State Genetic Algorithm (SSGA) based on tournament selection. SSGA is a non-generational evolutionary algorithm in which at each time step, individuals are selected for evaluation and reproduction, without a synchronized generational loop (see Lozano et al. (2008) for a survey).

The initial population in the "chemical GA" is also a collection of monomers, as in catalytic search. At every iteration, r individuals (the tournament size) are chosen at random and placed in a "catalyst pocket" C. The two best individuals (winners of the tournament) produce $r - 2$ children by crossover and mutation. These children replaced the other $r - 2$ individuals who had lost the tournament. The full algorithm is shown in Algorithm 2.

Note that in contrast with catalytic search, the GA is not mass-conserving: the new individuals might have completely different sizes from those they replaced. This is done in order to keep the chemical version of the GA as close as possible to a normal GA.

Algorithm 2 Steady State Genetic Algorithm in a Chemistry

```
1: $S$: multiset of candidate solutions
2: $r$: tournament size
3: $p_c$: crossover probability
4: $p_m$: mutation probability
5: **initialization:**
6: $S =$ random soup of $N$ monomers $m \in \Sigma$
7: **while** $true$ **do**
8:     $C$: set of tournament members
9:     expel $r$ random molecules out of $S$ and inject them
       into $C$
10:    expel the two fittest molecules $e_1$ and $e_2$ out of $C$
11:    **for** $i = 1$ to $r/2 - 1$ **do**
12:        **if** dice$(p_c)$ **then**
13:            $(p_1, p_2) \leftarrow$ crossover$(e_1, e_2)$
14:        **else**
15:            $p_1 = e_1, p_2 = e_2$
16:            **if** dice$(p_m)$ **then**
17:                $p_1 \leftarrow$ mutate$(p_1)$
18:            **end if**
19:            **if** dice$(p_m)$ **then**
20:                $p_2 \leftarrow$ mutate$(p_2)$
21:            **end if**
22:        **end if**
23:        inject $p_1$ and $p_2$ into $S$
24:    **end for**
25: **end while**
```

Experiments

Yamamoto (2010) compared catalytic search, GA and a random search to solve instances of the OneMax problem, extended to arbitrary target strings from a given alphabet Σ. The OneMax problem consists in maximizing the number of ones in a binary string, which is a special case of finding a hidden sentence $s \in \Sigma+$, made of a sequence of letters from Σ. This problem is known to be very easy to optimize, facilitating the comparison of the algorithms under ideal situations.

Yamamoto (2010) had already shown that catalytic search is able to solve simple problems, but in a slower manner than a GA. She had also shown that while catalytic search moves steadily towards the goal, a purely random search not only does not find the optimum but also diverges.

In this paper we focus on comparing catalytic search and GA under a changing environment, simulated by a target objective that is periodically modified. Furthermore, we investigate the influence of several parameters on the behavior of catalytic search, namely, the size of the enzyme pool, the amount of inflow/outflow, and the temperature. Two instances of the hidden sentence problem are used: one with binary strings with a target of all ones (OneMax), and another with an alphabetic sentence. They are shown in Table 1, where "id" is the identifier of the instance (subsequently labeled as "case 1" and "case 2" on the plots), and s_s is the size of the search space for each instance, when considering only sentences of length up to $|s|$.

id	Σ	$\|\Sigma\|$	target sentence s	$\|s\|$	s_s
1	01	2	1111111111111111	16	131070
2	a-z	26	catalyticsearch	15	1.744e+21

Table 1: Problem instances used

The ΔE_a cost function is set to the average length of the reacting strings, as in (Yamamoto (2010)). Fixed parameters set to default values are shown in Table 2.

size of the initial population of monomers	$N_0 = 100$
number of enzymes drawn from the enzyme pool for each catalysed reaction	$\|B\| = 10$
GA tournament size	$r = 4$

Table 2: Fixed parameter values

Results

We measure the obtained fitness and the ability to maintain diversity in the presence of changes. For catalytic search, we investigate the impact of the amount of inflow/outflow, the temperature and the size of the enzyme pool. Diversity is measured using a multiset diversity metric (Mattiussi et al. (2004)). It measures the fraction of unique elements (molecules) over the total size of the multiset (population size).

The target string changes 3 times during a run, at $t = 25, 50, 75$ (in units of 100 iterations). The target string is modified simply by applying the same mutation operator used in GA, with a given mutation probability per symbol of μ_t. All the results shown reflect the average of 10 runs.

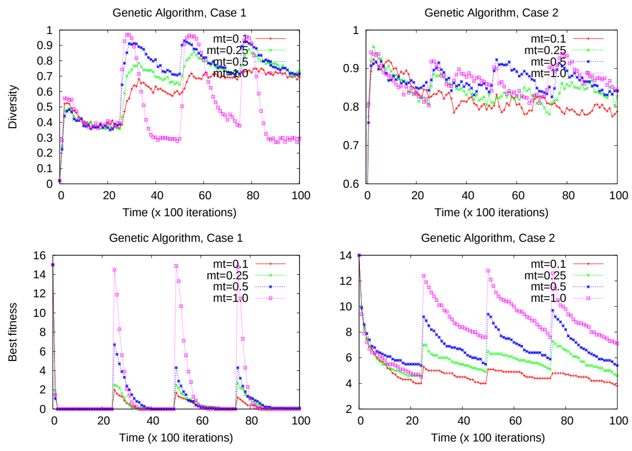

Figure 2: Average diversity and average best fitness for the genetic algorithm with changing target strings.

First of all, we compare GA and catalytic search for target mutation values μ_t varying from 0.1 to 1.0, representing slight to severe environmental changes.

Figure 2 shows the behavior of the GA under this scenario. As expected, bigger changes (represented by a higher μ_t) disturb the optimization process to a greater extent. For case 1, the amount of worsening in fitness corresponds roughly to the amount of mutation added. For example, for $\mu_t = 1.0$ (the target string changes entirely) the search restarts from scratch, with the best fitness jumping to 100% of its initial value at $t = 0$. For $\mu_t = 0.1$ (the target string changes slightly) the best fitness jumps to around 10% of its initial value, and so on. For case 2, the fitness also presents the characteristic sawtooth, but the recovery after changes is slower due to the higher difficulty of the problem.

The diversity of the population in GA displays a curious behavior under higher target mutation values. This is especially visible on case 1: soon after the target changes, the diversity jumps nearly to the maximum, and then decreases as the system approaches the optimum. The latter decrease in diversity is a well-known phenomenon in evolutionary computation, however the spontaneous jumps seem more surprising.

Figure 3 (left) shows the behavior of catalytic search under the same situation, for the case of no catalysis (empty enzyme pool), no inflow/outflow, and temperature $T = 1$. Naturally, the GA is much faster than the catalytic search at finding the optimum, which is an expected outcome. A

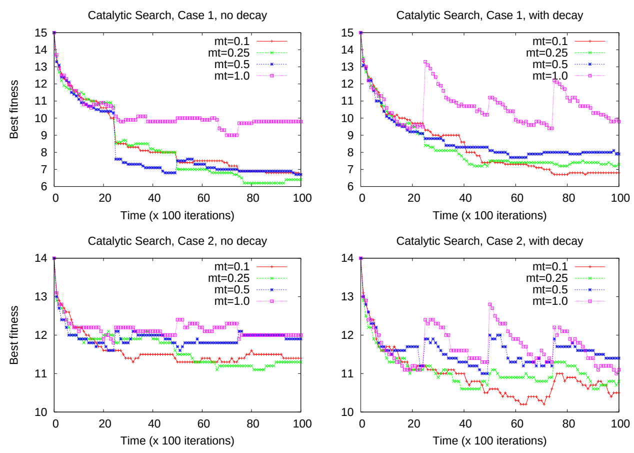

Figure 3: Average best fitness for catalytic search, with/without inflow/outflow.

more surprising result is that the behavior of catalytic search is qualitatively distinct from the GA: a small amount of target mutation does not seem to affect the system so clearly as it does for GA: sometimes, it even seems to help the search, such as around $t = 25$ for case 1 and $\mu_t \leq 0.5$.

Figure 3 (right) shows what happens when we introduce a small amount of inflow/outflow. This is represented by a decay parameter $p_d = 0.1$, meaning that at every iteration, with probability p_d, a negative tournament with size r is executed: $r = 4$ individuals are extracted at random from the population; their fitness is evaluated, and the one with the worst fitness (the loser of the tournament) is destroyed. It is then replaced by its length in new randomly generated monomers. In this way we ensure a mass-conserving inflow/outflow mechanism that combined with a negative selection mechanism makes sure that worse individuals are replaced with a higher probability. Here two types of behavior can be distinguished:

- for high target changes ($\mu_t \geq 0.5$) the behavior is qualitatively different from that with no inflow: it looks more like a GA (the fitness jumps when the target changes) although quantitatively (in terms of absolute fitness values) it still cannot optimize as fast as GA.
- for low target changes ($\mu_t \leq 0.25$) the behavior looks like the catalytic search with no inflow/outflow.

Increasing p_d does not seem to help: it floods the system with new monomers that cannot be consumed on time, and also causes the search to become more random.

Figure 4 compares the diversity of the population for catalytic search with and without inflow/outflow, for both cases studied. In contrast to the GA, the diversity in catalytic search is unaffected by the mutation of the target string. All mutation values produced similar figures, so we chose to plot only the results for $\mu_t = 0.5$.

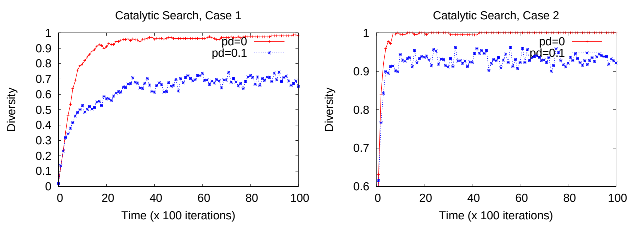

Figure 4: Average diversity for catalytic search, with and without inflow/outflow.

At the beginning, the population is made entirely of monomers, therefore the diversity is at most $|\Sigma|/N$, i.e. 0.02 for case 1 and 0.26 for case 2, for $N = 100$. It then increases progressively as new solutions are built by concatenating monomers. The fact that the diversity is close to the maximum for the case of no inflow/outflow ($p_d = 0$ on Fig. 4) means that in this situation, every individual in the population is nearly unique; there is no visible catalytic effect fostering the production of selected individuals.

For the case with inflow/outflow ($p_d = 0.1$ on Fig. 4) a lower diversity is observed. This is explained by the constant inflow of new monomers: since the size of the alphabet is small compared to the population, the monomer population necessarily contains a lot of copies of the same molecule. This is more evident for case 1, which uses a binary alphabet. There, the inflow causes the diversity to decrease much more prominently than in case 2.

Catalysis is expected to decrease diversity, by focusing the mass of the system into fewer species when the system is out of equilibrium. This phenomenon has not been observed in our system: the plots for $C_{max} = 100$ and $C_{max} = 1000$ closely resemble Fig. 4. This result indicates that the way catalysis is implemented in this system is not sufficient to modify the concentration pattern significantly when out of equilibrium, and focus most of the mass of the system into fewer, selected species. Catalysis does have a moderate effect on the performance, as will be shown in Figures 5 and 6. However, this effect is probably achieved primarily by accelerating a few reactions selectively by increasing their kinetic coefficients, and not by a significant concentration change. Even if faster, the enzymatic reactions do not succeed to focus sufficient mass, since the amount of possible reactions is not restricted: random crossover points are chosen at every time step, leading to different outcomes. This issue deserves further investigation. Actually, it is not straightforward to design an artificial chemistry to exhibit the focusing effect reported by Bagley and Farmer (1991), and it is even more difficult to cause it to spontaneously produce autocatalytic networks, which could later lead to the emergence of a GA-like scheme. On the other hand, the fact that catalytic search is able to keep diversity under a wide variety of conditions is a good property worth exploring.

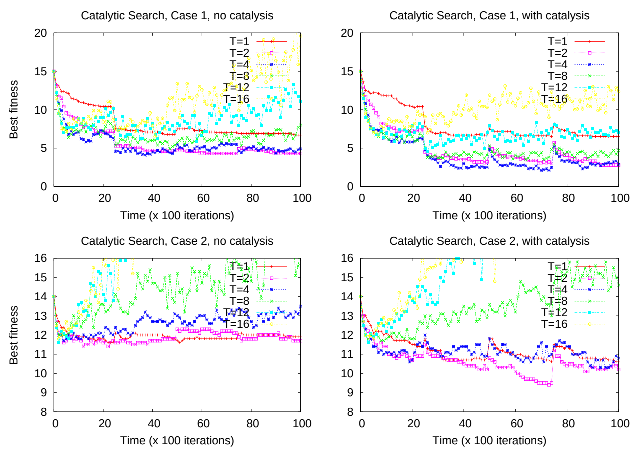

Figure 5: Influence of temperature and catalysis, no inflow/outflow

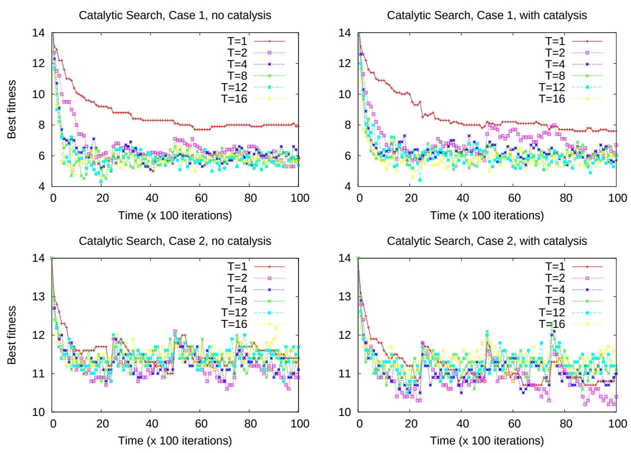

Figure 6: Influence of temperature and catalysis, with inflow/outflow

We now look at the influence of the temperature and of the amount of enzymes in the catalyst pool. We take $\mu_t = 0.5$ as an example (other values of μ_t produced similar results). The temperature makes all reactions faster, non-selectively, while the enzymes selectively speed up a few matching reactions. Figure 5 compares the best fitness of catalytic search for varying temperatures, with and without explicit catalysis, and no inflow/outflow ($p_d = 0$). We first look at the results without catalysis (left side). For case 1, increasing the temperature to moderate values improves the search: the optimum temperature is around $2 \leq T \leq 4$. For case 2, increasing the temperature does not seem to help: the best fitness does not improve. This can be explained by the fact that the energy barrier for case 1 might be too high, excessively penalizing the longer solutions necessary to solve this problem. Case 2 suffers from the same problem, but has a much larger search space, so merely increasing the temperature, a global parameter affecting all individuals, is not sufficient to improve the search.

Very high temperatures (for example, $T \geq 12$ for case 1, $T \geq 8$ for case 2, Fig. 5 (left), without catalysis) introduce more noise in the system, which becomes closer to a random search and hence tends to diverge.

Figure 5 (right) shows the effect of catalysts, for a total capacity of the catalyst pool set to $C_{max} = 1000$ enzymes. Catalysts help to improve the search and sometimes also help to stabilize the system: for lower temperatures, the system with catalysts moves faster towards the optimum; for higher temperatures, sometimes the catalysts prevent the search from becoming random, as for $T = 12$ in case 1.

When combining catalysis with inflow/outflow ($C_{max} = 1000$ and $p_d = 0.1$) the effect of catalysis becomes barely noticeable (Figure 6). This could be due to the fact that individuals that could be recognized by the enzymes are then selected for destruction, while new individuals for which there are no ready-made catalysts are created at a higher rate. Figure 6 also shows that the temperature has little impact on the performance (except for case 1 for $T = 1$ vs. other values of T). More importantly, the system with inflow/outflow no longer tends to diverge to a random search when the temperature increases, which is a positive aspect. The sawtooth pattern reminding us of GA appears here again, as in Figure 3 (right).

Related Work

This work was inspired mainly by Bagley and Farmer (1991), Banzhaf (1990), Kanada (1995), and Weeks and Stepney (2005).

Farmer et al. (1986) identify a critical probability of catalysis, near which the spontaneous emergence of self-sustaining autocatalytic networks becomes highly probable. Bagley and Farmer (1991) then show the spontaneous emergence of autocatalytic metabolisms, together with further conditions for their emergence. However, their results were based on a random assignment of catalytic efficiencies. Methods still lack for designing a proper structure-to-function mapping in a string-based chemistry, that would lead to a critical catalysis probability in the range needed for such emergent phenomenon to occur and persist. Hintze and Adami (2008) showed the evolution of metabolisms using a string-based chemistry with binding affinity and specificity. However, their design already assumes a whole cell structure with interacting genes and proteins.

Suzuki et al. (2003) enumerate minimal conditions for the evolution of artificial life forms, however they do so in a qualitative way. The quantitative conditions for the emergence of life subsystems (including metabolism) in an artificial environment are still not entirely understood, and methods for designing emergent algorithms based on these principles are still lacking. Designing algorithms inspired by such thin border between life and inanimate chemistry could help to understand such conditions and to devise corresponding methods in an iterative way.

The Molecular Travelling Salesman by Banzhaf (1990) is an optimization algorithm based on an artificial chemistry in which molecules representing candidate solutions are processed by machines that float in the reactor. These machines perform variation and selection, and are therefore closer to our version of GA in a chemistry.

In the Chemical Casting Model (CCM) by Kanada (1995), reaction rules modify and select molecules (candidate solutions) such as to drive the system towards a more ordered state (with lower entropy) in which molecules encode better solutions. The fitness mapping in CCM is similar to catalytic search: CCM seeks to maximize order by minimizing entropy (which is a macroscopic quantity), whereas catalytic search seeks to improve the fitness by moving towards lower energy levels at the microscopic level.

In the Artificial Catalysed Reaction Networks by Weeks and Stepney (2005), molecules encode partial solutions that are constructed via reversible polymerization reactions. Fitter products are rewarded by catalyzing their own production, therefore each molecule is potentially an autocatalyst, in contrast to our work where autocatalysis is not assumed.

A lot of work has been done on improving evolutionary computation for dynamic environments (see Jin and Branke (2005)). However, the potential of pre-evolutionary schemes in such context remains to be explored.

Summary and Conclusions

Our results reveal interesting aspects and point to many issues to be investigated. First of all, the behavior of catalytic search in the presence of changes is qualitatively different from that of an evolutionary algorithm. Evolution is capable of fast optimization, but is also more severely affected by changes. Catalytic search, on the other hand, is slower but also less sensitive to changes, and able to maintain a diverse pool of individuals in the population.

The behavior of catalytic search can be steered by parameters: a higher temperature, for instance, can cause the system to degenerate into a random search. Such degradation can be slowed down by the presence of catalysts, which have a stabilizing effect provided that the amount of inflow/outflow is very small or none.

Perhaps the most interesting phenomenon that could be expected from such a system would be a spontaneous transition to an autocatalytic or collectively autocatalytic stage, which could become a bridge towards a further transition to an evolutionary stage. So far however, we were not able to demonstrate such transitions in an emergent way. One of the major improvements needed in the current system is to ensure a larger impact of catalysts, in order to exhibit the focusing phenomenon that could enable such transitions to occur spontaneously. This would require a carefully designed structure-to-function mapping reflecting the required catalysis probabilities. It would also require a more efficient stochastic collision algorithm able to take into account a large number of possible reactions with rates differing by several orders of magnitude. Another major improvement needed is to make the system more tolerant to a continuous inflow/outflow, which is one of the primary conditions necessary for catalytic focusing to succeed.

References

Bagley, R. J. and Farmer, J. (1991). Spontaneous Emergence of a Metabolism. In *Artificial Life II*, pages 93–140. Addison-Wesley.

Banzhaf, W. (1990). The “Molecular” Traveling Salesman. *Biological Cybernetics*, 64:7–14.

Banzhaf, W., Dittrich, P., and Rauhe, H. (1996). Emergent Computation by Catalytic Reactions. *Nanotechnology*, 7:307–314.

Dittrich, P. (2005). Chemical Computing. In *Unconventional Programming Paradigms (UPP 2004), Springer LNCS 3566*, pages 19–32.

Dittrich, P. and Banzhaf, W. (1998). Self-Evolution in a Constructive Binary String System. *Artificial Life*, 4:203–220.

Farmer, J. D., Kauffman, S. A., and Packard, N. H. (1986). Autocatalytic replication of polymers. *Physica D*, 2(1-3):50–67.

Fontana, W. and Buss, L. W. (1994). ‘The Arrival of the Fittest’: Toward a Theory of Biological Organization. *Bulletin of Mathematical Biology*, 56:1–64.

Hintze, A. and Adami, C. (2008). Evolution of complex modular biological networks. *PLoS Comput Biol*, 4(2):e23.

Jin, Y. and Branke, J. (2005). Evolutionary Optimization in Uncertain Environments - A Survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303–317.

Kanada, Y. (1995). Combinatorial Problem Solving Using Randomized Dynamic Composition of Production Rules. In *IEEE International Conference on Evolutionary Computation*, pages 467–472.

Lozano, M., Herrera, F., and Cano, J. R. (2008). Replacement Strategies to Preserve Useful Diversity in Steady-State Genetic Algorithms. *Information Sciences*, 178(23):4421–4433.

Mattiussi, C., Waibel, M., and Floreano, D. (2004). Measures of Diversity for Populations and Distances Between Individuals with Highly Reorganizable Genomes. *Evolutionary Computation*, 12(4):495–515.

Nowak, M. A. and Ohtsuki, H. (2008). Prevolutionary Dynamics and the Origin of Evolution. *PNAS*, 105(39).

Suzuki, H., Ono, N., and Yuta, K. (2003). Several necessary conditions for the evolution of complex forms of lif e in an artificial environment. *Artificial Life*, 9(2):153–174.

Weeks, A. and Stepney, S. (2005). Artificial Catalysed Reaction Networks for Search. In *ECAL Workshop on Artificial Chemistry*.

Yamamoto, L. (2010). Evaluation of a Catalytic Search Algorithm. In *Proc. 4th International Workshop on Nature Inspired Cooperative Strategies for Optimization (NICSO)*, Granada, Spain.