# Evaluation of a Catalytic Search Algorithm

Lidia Yamamoto

**Abstract** We investigate the search properties of pre-evolutionary random catalytic reaction networks, where reactions might be reversible, and replication is not taken for granted. Since it counts only on slow growth rates and weak selective pressure to steer the search process, catalytic search is an inherently slow process. However it presents interesting properties worth exploring, such as the potential to steer the computation flow towards good solutions, and to prevent premature convergence. We have designed a simple catalytic search algorithm, in order to assess its beamed search ability. In this paper we report preliminary results that show that although weak, the search strength achieved with catalytic search is sufficient to solve simple problems, and to find good approximations for more complex problems, while keeping a diversity of solutions and their building blocks in the population.

## 1 Introduction

Artificial Chemistries have the ability not only to *model* evolutionary behavior but also to *create* it, or to cause it to *emerge* spontaneously [3, 9–11]. However, the exact conditions upon which such evolutionary behaviour could emerge are not entirely clear, and are deeply linked to the conditions for the transition from inanimate to

Lidia Yamamoto
Computer Science Department, University of Basel
Bernoullistrasse 16, CH-4056, Basel, Switzerland
e-mail: Lidia.Yamamoto@unibas.ch

pre-print of paper to appear in:
Proceedings of the 4th. International Workshop on Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), May 12-14, 2010, Granada, Spain.

living matter. Another aspect that remains still unclear so far is how to harness the *emergent computation* [7, 12] properties of such chemistries for the construction of beamed search schemes able to optimize solutions to user-defined problems.

A number of chemically-inspired approaches to optimization towards user-defined goals have been proposed [5, 6, 14, 20, 22]. The reaction networks created by such chemistries may exhibit complex dynamics, hence the general problem of searching with a chemistry remains poorly understood.

Some chemistries take evolution elements for granted, such as replication, therefore the problem of how to get evolutionary behavior is not an issue for them. In contrast, in this paper we look at the particular case of chemistries that do not assume replication, and that must comply to some physical laws such as mass and energy conservation. The behavior of such chemical search can be classified as *pre-evolutionary* [18]. Chemical reactions consume educts to produce new molecules, in a mass-conserving way, and most reactions are reversible. Catalysts may be present to enhance the rate of some reactions. The resulting mechanism is a *Catalytic Search*, that relies only on slow growth rates and weak selective pressure to steer the search process.

Catalytic search is an inherently slow process in general, but after some time it could reach an *autocatalytic* stage where some elements of the network become able to replicate directly or collectively via cooperative interactions. The question that remains unanswered is whether such a slow process is sufficient to ignite a faster, more efficient search process exhibiting full Darwinian evolutionary dynamics within feasible runtimes, and if yes, how this could be achieved.

Although typically slow, catalytic search has a useful potential as a "soft search" mechanism, which remains under-explored so far. As pointed out in [22], catalytic search presents interesting properties worth exploring, such as the potential to undo wrong computations through reversible reactions, to steer the flow of the system 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. Moreover catalytic search is inherently *cooperative*: since molecules cannot self-replicate in principle, they need the help of other molecules in order to grow. Hence they are forced to self-organize into a network of positive interactions that construct and deconstruct solutions dynamically, according to the objective function to be computed.

We have designed a simple catalytic search algorithm, in order to assess the ability of a catalytic artificial chemistry subject to pre-evolutionary dynamics to exhibit a beamed search behavior. The chemical search scheme is built on top of a thermodynamic model which steers the candidate solutions not only towards better fitness but also towards lower computation costs. We compare Catalytic Search with a pure random search (for the sake of sanity check only), and with a variation of a Steady-State Genetic Algorithm (SSGA) [16] implemented with an artificial chemistry. As expected, the performance of the catalytic search scheme lies between that of a pure random search and that of plain evolutionary search represented by the genetic algorithm. However catalytic search presents other interesting properties, such as the preservation of diversity and of partial solution building blocks in the population.

The paper is organized as follows: Section 2 surveys the related literature on search schemes based on chemistry. Section 3 describes our catalytic search algorithm, and report some experimental results on the simple problem of finding a hidden sentence. Section 4 concludes with an outlook on the many interesting avenues to explore.

## 2 Background

A survey of optimization schemes based on artificial chemistries can be found in Section 4.3 of [10]. Here we summarize and update it.

Chemical approaches to optimization towards a user-defined goal have been proposed in [5, 6, 14, 20, 22]. These approaches can be divided into two categories: In the first category we find search algorithms inspired by chemistry, but for which the actual solutions searched are not encoded as chemical computing programs but as parameters to be optimized [5, 14], as partial solutions to the problem [22], or as conventional program trees [20]. In the second category we find centralized evolutionary algorithms used to evolve chemical reaction networks by manipulating their graphs [8], or to evolve chemical computing programs by genetic programming [6]. The work reported in this paper falls into the first category. However our long term goal is to combine both categories in one, obtaining a search process based on chemistry, for searching solutions encoded as chemical programs.

An optimization method inspired by chemistry is presented in [5]. Candidate solutions are encoded as strings analogous to macromolecules in prebiotic evolution. These strings carry a quality signal (fitness value). Machines (analogous to enzymes) operate on the strings to change and select them according to a fitness function embedded within the machine's knowledge.

The Chemical Casting Model (CCM) [14] is inspired by the process of entropy reduction which is behind many self-organization phenomena in chemistry. Candidate solutions are encoded as molecules; reaction rules modify and select molecules, driving the system towards a more ordered state (with lower entropy) in which molecules encode better solutions. CCM has been successfully applied to many different problems ranging from constraint satisfaction to graph coloring and the traveling salesperson.

Chemical Genetic Programming [20] takes inspiration from gene expression and chemistry in order to construct program trees in Genetic Programming.

ARMS (Abstract Rewriting Systems on Multisets [21]) is a chemical evolution system based on Membrane Computing or P Systems [19]. Membrane Computing allows hierarchies of multiset compartments to be constructed recursively. ARMS makes use of this feature to evolve populations of artificial cells by a process of cell growth and division. The resulting cells may exhibit a rich internal structure, sometimes resembling protocells models such as the chemoton [13]. The ARMS system has been applied to the evolution of artificial cells both for biological and for computational purposes.

Our work builds upon the Artificial Catalysed Reaction Networks from [22]. That scheme takes inspiration from Kaufffman's autocatalytic networks [4, 15]. Each molecule is a partial solution. The algorithm starts with a population of small molecules and builds larger ones via polymerization reactions. Fitter products are rewarded by catalyzing their own production. Each molecule is therefore an autocatalyst. Compared with [22], in our work we do not assume that molecules are autocatalysts, and we use a crossover operator that includes polymerization as a special case.

## 3 Catalytic Search Algorithm

The Catalytic Search Algorithm works as follows: initially, a random soup of molecules is generated. Each molecule is a candidate solution represented as a string of symbols from an alphabet $\Sigma$. At every time step, two molecules are chosen for collision. They react with a probability $k_f$, which maps to 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 collision is elastic with probability $(1-k_f)$, in which case the molecules are put back into the soup and no products are generated.

A crossover reaction can be written as follows:

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

Here is an example, for strings from an alphabet $\Sigma = \{a,b,c\}$:

$$abba + ccb \underset{k_r}{\overset{k_f}{\rightleftharpoons}} abbc + acb \tag{2}$$

Crossover is a mass-conserving operation, i.e. it conserves the total number of symbols before and after the reaction.

The initial population is always a soup of monomers (strings of length one). Solutions are then built by concatenating these monomers. This is a special case of a crossover operation, where the crossover point on one of the strings is the end of the string, and the crossover point on the other string is at the beginning. More complex solutions can then be constructed out of these basic building blocks.

We choose the coefficients $k_f$ and $k_r$ to be a function of the fitness and the computation cost associated with the solution, in order to steer the search by differentiated reaction rates. This mapping will be explained below.

Once the molecules have collided, the reaction only occurs if the molecules have sufficient kinetic energy in order to overcome the *activation energy barrier* ($E_a$), which corresponds to the minimum amount of kinetic energy that is needed for the reaction to occur.

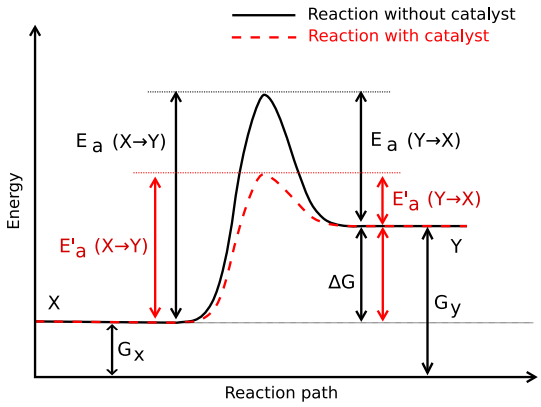


**Fig. 1** Potential energy changes during catalysed and uncatalysed chemical reactions. Figure adapted from [1].

Figure 1 shows a typical plot of the potential energy changes during a chemical reaction. The horizontal axis is called *reaction coordinate*, and shows the progression of the (forward) reaction from reactants $X$ on the left side to products $Y$ on the right (symmetrically, the corresponding reverse reaction can be read from right to left). The vertical axis shows the corresponding potential energy. The height of the peaks with respect to the initial state corresponds to the *activation energy* of 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. The difference in potential energy before and after the reaction is given by $\Delta G$:

$$\Delta G = G_p - G_e \tag{3}$$

If $\Delta G > 0$ then the reaction is *endergonic*, i.e. it absorbs energy from its surroundings, while if $\Delta G < 0$ it is *exergonic*, releasing energy. Endergonic reactions are typically non-spontaneous, i.e. their equilibrium is shifted towards the educts, while exergonic reactions occur typically spontaneously, resulting in larger quantities of products.

In order to steer the system towards the production of fitter solutions, we map the fitness of the solution to the potential energy of its molecule. A lower value of the fitness function is often associated with a better fitness, for instance, a shorter distance to the optimum. In this case, we can associate fitness with the the *potential energy* of the molecule directly. The total potential energy of the educts $G_e$ (resp. products $G_p$) is the sum of potential energies of each educt (resp. product) involved in the reaction, i.e. the sum of their fitness values. In this way, the production of fitter solutions (i.e. with lower potential energy) is spontaneous, while the production of poor solutions is non-spontaneous.

In order to provide the system with an incentive for efficient computations, we further map the activation energy for a reaction to the estimated computation cost of producing a solution. For instance, let us take a simple case in which the cost is a linear function of the length of the candidate solutions. Since we only consider mass-conserving (i.e. symbol-conserving) operations, the total number of atoms is the same on both educt and product sides. An increase in activation energy $\Delta E_a$ is then added on top of the highest potential energy $G$. $\Delta E_a$ corresponds to the portion $E_a(Y \rightarrow X)$ in Figure 1. As a result, the side of the reaction with the lowest potential energy (the $X$ side to the left of Figure 1) will see an activation energy of $E_a = \Delta E_a + |\Delta G|$, while the other side ($Y$, on the right) will see $E_a = \Delta E_a$. The portion $\Delta E_a$ of the total activation energy is set to the average length of the educts (or products):

$$\Delta E_a = \frac{|A| + |B|}{2} = \frac{|C| + |D|}{2} \tag{4}$$

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

$$\text{if} \quad \Delta G \leq 0 \quad \begin{cases} E_{af} = \Delta E_a \\ E_{ar} = \Delta E_a - \Delta G \end{cases} \tag{5}$$

$$\text{if} \quad \Delta G > 0 \quad \begin{cases} E_{af} = \Delta E_a + \Delta G \\ E_{ar} = \Delta E_a \end{cases} \tag{6}$$

The coefficient $k_f$ (resp. $k_r$) is determined as a function of the activation energy, following the *Arrhenius equation* from chemistry [2]:

$$k = Ae^{-\frac{E_a}{RT}} \tag{7}$$

where $A$ is the so-called *pre-exponential factor* of the reaction, $E_a$ is its *activation energy*, and $RT$ are constants. In our case, we set $A = 1$ and $\beta = \frac{1}{RT}$ is a configuration parameter of our algorithm (currently set to $\beta = 1$).

The constants $k_f$ (resp. $k_r$) determine the probability that the reaction is successful once the reactants collide. According to the Arrhenius equation (Eq. 7), these coefficients decrease exponentially with the activation energy barrier $E_a$ seen by the reactants. Since $E_a$ increases with $\Delta E_a$, which is mapped to the computational cost of the operation, the probability of the reaction to occur decreases with its cost, as desired. Similarly, since the height of the $E_a$ barrier observed is higher on the side of the reaction with lower $G$, it is more difficult to "cross the barrier" from this side, therefore it is more difficult to move from a lower (closer to the optimum) to a higher (farther from the optimum) fitness value, which is also the behavior that we are seeking. While lower, there is still some probability to move towards worse solutions, since that may help creating new solutions which might be useful for the search.

In this way, this scheme is able to steer the flow of production of candidate solutions towards better ones. There is no explicit replication, and no 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.

**Algorithm 1** Catalytic Search Algorithm

$T$: maximum number of iterations
$0 \leq t < T$: current iteration
$S$: multiset of candidate solutions
$C$: pool of enzymes (catalysts) of maximum capacity $C_{max}$
**initialization:**
$t = 0$
$S =$ random soup of $N$ monomers $m \in \Sigma$
$C = \emptyset$
**while** $t < T$ and solution not found **do**
    expel two random molecules $e_1$ and $e_2$ out of $S$
    $(i_1, i_2) =$ random crossover points within $e_1$ and $e_2$
    $(p_1, p_2) \leftarrow$ crossover$(e_1, e_2, i_1, i_2)$
    $G_e =$ fitness$(e_1)+$ fitness$(e_2)$
    $G_p =$ fitness$(p_1)+$ fitness$(p_2)$
    $\Delta G = G_p - G_e$
    $E_a = (|e_1| + |e_2|)/2$
    **if** $\Delta G > 0$ **then**
        $E_a \leftarrow E_a + \Delta G$
    **else if** $\Delta G < 0$ **then**
        $c =$ "crossover$(e_1, e_2, i_1, i_2)$": the enzyme that catalyses this reaction
        $n_c =$ multiplicity of $c$ in $C$
        **if** $n_c > 0$ **then**
            $E_a \leftarrow E_a / n_c$
        **end if**
        $p_c = |\Delta G| / |\Delta G_{max}|$
        add another instance of $c$ to $C$ with probability $p_c$
        **while** $|C| > C_{max}$ **do**
            destroy a random catalyst from $C$
        **end while**
    **end if**
    $k_f = e^{-\beta E_a}$
    **if** dice$(k_f)$ **then**
        inject new products $p_1$ and $p_2$ into $S$
    **else**
        inject educts $e_1$ and $e_2$ back to $S$
    **end if**
    $t \leftarrow t + 1$
**end while**

## *3.1 Catalysts*

The above scheme is able to steer the search process, but in a weak way. In order to improve steering and to make the search more beamed, enzymes that catalyse

the reactions can be included. Enzymes decrease the activation energy necessary for the reaction, as depicted in Figure 1. They do so on both forward and reverse sides of the reaction, therefore the equilibrium concentrations do not change. However, as shown in [4], under some conditions, catalysts can focus the reaction network into a few species, creating a selection pressure towards a metabolic core. One of the conditions for obtaining such *catalytic focusing* is that the system is kept out of equilibrium by an inflow of food material.

Here we introduce a simpler kind of catalyst which is not entirely faithful to chemistry, as it will work to reduce the activation energy barrier, but only in the direction of fitness improvement. We have temporarily adopted such annoying violation of the chemical laws because our first experiments have shown that maintaining the system out of equilibrium for such an optimization purpose is not such an easy task: in order to keep the system within a reasonable mass balance, an inflow of material (e.g. monomers) requires a corresponding outflow of other, potentially more complex solution molecules. If we remove such molecules at random, we might lose important partial solutions. Since the system is slow to replenish them, the optimization process is hindered. If we remove worse fit molecules with a higher probability, then the equilibrium is shifted towards the production of more of such bad molecules. If we do the opposite, i.e. remove the fitter molecules, then the system will tend to replenish them, but too slowly. Similar problems are reported in [22]. A good method for keeping the system out of equilibrium without disrupting the search process is still lacking. This topic deserves further investigation.

Our catalysts are strings of the form: "$op(s_1, s_2, p_1, p_2)$", where $op$ is an operator (currently only the crossover operator is supported), $s_1$ and $s_2$ are the educt strings, $p1$ and $p_2$ are parameters indicating the crossover points in $s_1$ and $s_2$ respectively.

When two molecules collide, it is checked whether they have one or more matching catalysts. If matching catalysts are found, they will be used to increase the reaction probability, as explained below. Currently only exact match is supported. In the future, enzymes could bind to their substrates with a certain affinity, proportional to how well their strings match, for instance by using a distance metric such as the Hamming distance or a string alignment algorithm such as the edit distance.

The complete algorithm is shown in Algorithm 1. 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 will facilitate their reaction, by lowering the corresponding $\Delta E_a$, which then becomes:

$$\Delta E_a' = \frac{\Delta E_a}{n_c} \tag{8}$$

where $\Delta E_a$ is calculated according to Equation (4), and $n_c$ is the concentration (multiplicity) of the corresponding catalyst in the catalyst pool.

### 3.2 Find the Hidden Sentence

We compare Catalytic Search with a pure random search (for the sake of sanity check only), and a variation of a Steady-State Genetic Algorithm (SSGA) based on a tournament selection mechanism implemented using an artificial chemistry. SSGA [16] is a non-generational evolutionary algorithm in which at each time step, individuals are selected for evaluation and reproduction, without a synchronized generational loop.

The three algorithms have been applied to the simple problem of finding a hidden string. In [22] Catalytic Search is applied to the OneMax problem, which consists in maximizing the number of ones in a binary string. This is a special case of finding a hidden string, i.e. a sentence $\Sigma^+$ made of a sequence of letters from an alphabet $\Sigma$. The length of the sentence can be variable, and the algorithm does not know anything about the nature of the solution. It is guided only by the fitness, given as the distance from the optimum. This problem has a smooth fitness landscape with a unique peak, and is therefore easy to optimize.

The fitness function for this problem is simply:

$$f(i) = d(i, i^*) \tag{9}$$

where $d(i, i^*)$ is the distance between the candidate solution $i$ and the target sentence $i^*$. The function $d(i, j)$ is taken as the edit distance between the two strings, i.e. the minimum number of edit operations (add, delete, replace symbol) necessary to convert one string into the other. The best fitness value is thus the smallest distance, i.e. $d(i, j) = 0$.

### 3.3 Experimental Results

We have simulated the three algorithms on simple cases, and show a few preliminary results in this section. For each of the three algorithms three test cases have been performed, according to Table 1, where $L$ is the length of the target solution $i^*$, and $s$ is the size of the search space for each case, for sentences of length up to $L$.

| case n. | alphabet $\Sigma$ | target solution $i^*$ | length $L = \|i^*\|$ | search space $s$ |
|---|---|---|---|---|
| 0 | ABCD | AABBCCDD | 6 | 87380 |
| 1 | 01 | 1111111111111111 | 16 | 131070 |
| 2 | a-z | thisisatest | 11 | 3.81716e+15 |

**Table 1** Test cases

For each algorithm, 100 runs were performed. The genetic algorithm was run with a tournament of size $r = 4$, a mutation probability of $p_m = 0.1$ and a crossover

probability of $p_c = 0.9$. The population size was $N = 100$ molecules for all the algorithms and cases, and the maximum number of iterations was $T = 10000$.

Table 2 shows the number of exact solutions found, per test case and per algorithm.

| case n. | random search | catalytic search | genetic algorithm |
|---|---|---|---|
| 0 | 0 | 38 | 75 |
| 1 | 1 | 0 | 100 |
| 2 | 0 | 0 | 3 |

**Table 2** Number of exact solutions found, out of 100 runs per algorithm per test case.

As expected, the genetic algorithm is able to find a higher number of exact solutions due to its stronger replication and selective pressure. Also as expected, random search performed very poorly. Catalytic search was only able to find a significant amount of exact solutions on the first, easier case. The best solutions found in other cases were approaching the optimum but very slowly. We can see this in Fig. 2 (left), which shows the average best fitness for case 2, per algorithm. In Fig. 2 (top left) we can see that random search not only does not make progress, but diverges to worse solutions. The catalytic search (Fig. 2 (middle left)), although not entirely optimal, displays a qualitative behavior that is similar to the genetic algorithm, showing steady progress towards the optimum.

The diversity of the population has been measured using a multiset diversity metric [17]. It measures the fraction of unique elements (molecules) over the total size of the multiset (population size). Although it rises almost to the maximum for the catalytic search scheme (Fig. 2 top right), the system does not get "lost", and still displays a well-behaved search towards the solution. In comparison, the diversity in the genetic algorithm goes up at the beginning, and then drops to a middle level, as the system approaches the optimum (not shown).

The solutions in catalytic search show modest elongation towards the optimum length ($L = 11$ for case 2) (Fig. 2 middle right), while for the same case the genetic algorithm quickly moves towards solutions that are longer than the optimum (Fig. 2 bottom right). Catalytic search conserves the number of atoms, while the genetic algorithm produces an increasing number of atoms at the beginning, and this number then slowly drops and then stabilizes as the optimum is approached.

Similar qualitative behaviors have been observed for the other test cases from Table 1 (not shown).

## 4 Conclusions and Future Work

Catalytic search illustrates that optimization is possible even in the absence of explicit Darwinian selection. The selection force here is much weaker, progress is slower, and the systems not always converges to the optimum. Such a search method

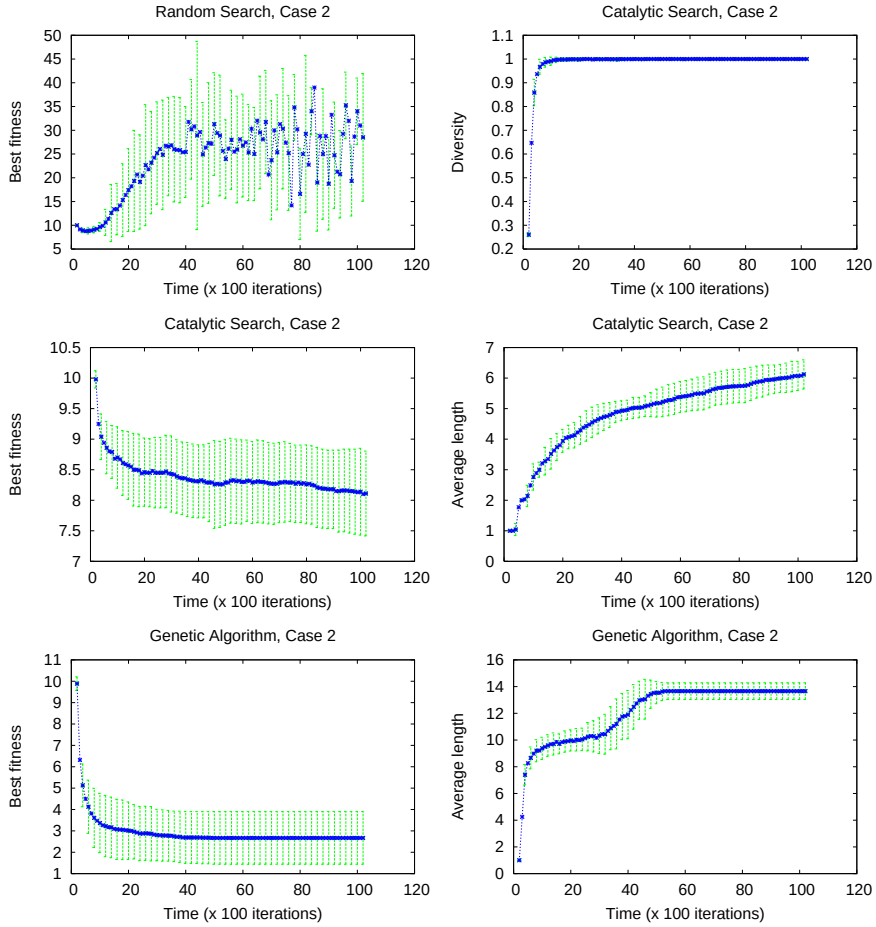


**Fig. 2** Experimental results comparing different search schemes. Average values over 100 runs, with errorbars indicating the standard deviation. Left: Average best fitness for each algorithm. Top right: diversity of the population for the catalytic search. Middle right: Average length of the solutions in catalytic search. Bottom right: Average length for the GA.

is inherently suboptimal, and not intended as a replacement for evolutionary algorithms or other successfully established heuristic search methods. As shown in models of pre-evolutionary dynamics [18], a prelife model with no established Darwinian evolution properties can be invaded as soon as self-replicants cross an efficiency threshold. In the optimization domain, catalytic search relates to prelife as genetic algorithms relate to Darwinian evolution. However, in the same way as prelife played a crucial role towards life, catalytic search can play a role as a "soft" search method, in a more exploratory phase of the search. It might prove useful in dynamic or noisy environments, to let a variety of solutions survive, to dampen temporary fluctuations in input parameters, and to undo or revert to past states when

needed. We believe that there is a potential that remains to be explored in such soft search schemes, although we are not able to show this entire potential here to its full extent. We were able to show some properties such as an apparent ability to keep a higher diversity of solutions in the population without any explicit diversity maintenance mechanism. Other properties described in [22] remain to be demonstrated. We are particularly interested in the potential to undo wrong computations via reversible reactions, and to steer the flow of computation using an open system driven out of equilibrium as in [4]. This is difficult to achieve in a search algorithm, due to the risk of flushing out good solutions or their building blocks.

Many points remain to be improved in our current implementation: The catalysis model should support affinity matching. Catalysts should be inserted in the same pool together with the candidate solutions, and the reaction algorithm should model collisions involving catalysts and substrates explicitly. Furthermore, the saturation of enzymes must be considered, moving from mass action to enzyme kinetics. A more accurate diversity metric should be considered, taking into account the distance between strings. An analysis of the topology of catalytic networks should be undertaken, in order to detect potential autocatalytic sets, and search for emergent feedback loops and collective replicators. The main unsolved issue so far is to find a good way to keep the system out of equilibrium and yet in a focused optimizing mode.

**Acknowledgements** This work has been supported by the European Union through FET Project BIONETS. The author would also like to thank Thomas Meyer, Wolfgang Banzhaf, and the anonymous reviewers for their helpful comments and encouragement.

## References

[1] Activation Energy, Wikipedia (2006) Activation Energy, Wikipedia, http://en.wikipedia.org/wiki/Activation_energy
[2] Atkins P, de Paula J (2002) Physical Chemistry. Oxford University Press
[3] Bagley R, Farmer J, Fontana W (1991) Evolution of a Metabolism. In: Artificial Life II, Addison-Wesley, pp 141–158
[4] Bagley RJ, Farmer J (1991) Spontaneous Emergence of a Metabolism. In: Artificial Life II, Addison-Wesley, pp 93–140
[5] Banzhaf W (1990) The "Molecular" Traveling Salesman. Biological Cybernetics 64:7–14
[6] Banzhaf W, Lasarczyk C (2004) Genetic Programming of an Algorithmic Chemistry. In: O'Reilly, et al (eds) Genetic Programming Theory and Practice II, vol 8, Kluwer/Springer, chap 11, pp 175–190
[7] Banzhaf W, Dittrich P, Rauhe H (1996) Emergent Computation by Catalytic Reactions. Nanotechnology 7:307–314
[8] Deckard A, Sauro HM (2004) Preliminary Studies on the In Silico Evolution of Biochemical Networks. ChemBioChem 5(10):1423–1431

[9] Dittrich P, Banzhaf W (1909) Self-Evolution in a Constructive Binary String System. Artificial Life 4:203–220
[10] Dittrich P, Ziegler J, Banzhaf W (2001) Artificial Chemistries – A Review. Artificial Life 7(3):225–275
[11] Fontana W, Buss LW (1994) 'The Arrival of the Fittest': Toward a Theory of Biological Organization. Bulletin of Mathematical Biology 56:1–64
[12] Forrest S (1990) Emergent Computation: Self-organizing, Collective, and Cooperative Phenomena in Natural and Artificial Computing Networks. Physica D 42(1-3):1–11
[13] Gánti T (2003) Chemoton Theory, Volume 1: Theoretical Foundations of Fluid Machineries. Kluwer Academic
[14] Kanada Y (1995) Combinatorial Problem Solving Using Randomized Dynamic Composition of Production Rules. In: IEEE International Conference on Evolutionary Computation, pp 467–472
[15] Kauffman SA (1993) The Origins of Order: Self-Organization and Selection in Evolution. Oxford University Press
[16] Lozano M, Herrera F, Cano JR (2008) Replacement Strategies to Preserve Useful Diversity in Steady-State Genetic Algorithms. Information Sciences 178(23):4421–4433
[17] Mattiussi C, Waibel M, Floreano D (2004) Measures of Diversity for Populations and Distances Between Individuals with Highly Reorganizable Genomes. Evolutionary Computation 12(4):495–515
[18] Nowak MA, Ohtsuki H (2008) Prevolutionary Dynamics and the Origin of Evolution. PNAS 105(39)
[19] Paun G (2000) Computing with Membranes. Journal of Computer and System Sciences 61(1):108–143
[20] Piaseczny W, Suzuki H, Sawai H (2004) Chemical Genetic Programming - Evolution of Amino Acid Rewriting Rules Used for Genotype-Phenotype Translation. In: Congress on Evolutionary Computation (CEC), vol 2, pp 1639–1646
[21] Suzuki Y, Fujiwara Y, Takabayashi J, Tanaka H (2001) Artificial Life Applications of a Class of P Systems: Abstract Rewriting Systems on Multisets. In: Workshop on Multiset Processing (WMP), Springer, London, UK, pp 299–346
[22] Weeks A, Stepney S (2005) Artificial Catalysed Reaction Networks for Search. In: ECAL Workshop on Artificial Chemistry