Maximally Distant Codes Allocation Using Chemical Reaction Optimization with Enhanced Exploration

Taisir Eldos
Department of Computer Engineering
Jordan University of Science and Technology
Irbid, Jordan

Abdallah Khreishah
Department of Electrical and Computer Engineering
New Jersey Institute of Technology
Newark, NJ – USA

***Abstract*—Error correcting codes, also known as error controlling codes, are sets of codes with redundancy that provides for error detection and correction, for fault tolerant operations like data transmission over noisy channels or data retention using storage media with possible physical defects. The challenge is to find a set of m codes out of 2^n available n-bit combinations, such that the aggregate hamming distance among those codewords and/or the minimum distance is maximized. Due to the prohibitively large solution spaces of practically sized problems, greedy algorithms are used to generate quick and dirty solutions. However, modern evolutionary search techniques like genetic algorithms, swarm particles, gravitational search, and others, offer more feasible solutions, yielding near optimal solutions in exchange for some computational time. The Chemical Reaction Optimization (CRO), which is inspired by the molecular reactions towards a minimal energy state, emerged recently as an efficient optimization technique. However, like the other techniques, its internal dynamics are hard to control towards convergence, yielding poor performance in many situations. In this research, we proposed an enhanced exploration strategy to overcome this problem, and compared it with the standard threshold based exploration strategy in solving the maximally distant codes allocation problem. Test results showed that the enhancement provided better performance on most metrics.**

Keywords*—Evolutionary Algorithms; Chemical Reaction Optimization; Maximally Distant Codes; Binary Knapsack Problem; Fault Tolerance

I. INTRODUCTION

The goal of this research is to allocate sets of codes that maximize mutual distances for use as error control codes. This is of great significance for many applications, like data retrieval and communication. Finding optimal or even near optimal solutions for practically sized problems using brute force search is a challenge due to the prohibitively large solution spaces. The problem A(n, m, d) is about locating a set of m codewords with n bits that are at least d bit apart. The solution space of the small instance (7, 16, 3) is at least 1020, while a slightly larger instance like (8, 16, 3) has more than 1024 solutions to explore. The solution space of a relatively small sized problem, like (10, 64, 3), exceeds 480 Googol (4.8x10102), which is large enough to rule out any exact search algorithm even using supercomputing power.

According to the packing sphere theorem, aka Hamming bound, if $\boldsymbol{S}$ is a code of strings of $\boldsymbol{n}$ bits with $\boldsymbol{d(S) = 2k+1}$, where $\boldsymbol{k}$ is the radius (maximum number of bits that can be corrected) and $\boldsymbol{d}$ is the distance, then the cardinality $\boldsymbol{|S|}$, or size set, is defined as:

$$|S| \leq \frac{2^n}{\sum_{i=0}^{k} C(n,\ i)} \tag{1}$$

Using (1), loose bounds on the sets cardinality are shown in Table I for selected values of $\boldsymbol{d}$ and $\boldsymbol{k}$; $\boldsymbol{d}$ errors detection and $\boldsymbol{k}$ errors correction. If the inequality in (1) does not hold then the code $\boldsymbol{S}$ does not exist while if it holds, there is no assurance of existence of such a code, and this is why those bounds are not quite useful and better ones are needed, later sections will provide tighter ones as research outcomes. Later, tighter bounds will be presented as reported by researchers.

TABLE I. CODEWORDS VERSUS MINIMAL DISTANCE AND RADIUS

	Error Correction / Detection		
n	***k = 1, d = 3***	***k = 2, d = 5***	***k = 3, d = 7***
8	28	6	2
10	93	18	5
12	315	51	13
14	1092	154	34
16	3855	478	94
18	13797	1524	265

Evolutionary optimization algorithms offer optimal or near optimal solutions with affordable computational effort; time and resources. Such algorithms have been used in solving complex engineering problems with varying quality and time tradeoffs. The CRO algorithm has emerged recently as an adaptive method to explore such large spaces with ordinary computational resources. In a previous work, we adapted the CRO algorithm to the maximally distant codes allocation problem by mapping it to the well-known binary Knapsack problem. The results were acceptable but with poor exploitation and exploration balance in some situations, as the algorithm spent more time exploring the space even with tight exploration reactions conditions, yielding less fruitful computations. We extended the research to adjust the exploration reactions through better control; randomized and quality dependent rather than threshold only dependent.

II. LITERATURE REVIEW

Due to significance, error correcting code allocation problem has been addressed for a long time using various methods. Evolutionary paradigms were applied with varying degrees of success; for example, the work in [1] described some initial experiments of Genetic Algorithms (GAs) to discover maximal distance codes, and discussed the potential advantage of genetic algorithms in this problem domain, others like [2], compared the performance of many evolutionary algorithms with local search and greedy methods in solving the error correcting codes discovery problem, and concluded that the GAs were the best of all other algorithms in general, with even more performance advantage as the cases got harder, while [3] tackled the error correcting codes allocation with two related techniques, Memetic Algorithm (MA) and Scatter Search (SS), by investigating the instantiation of those techniques for error correction codes design, the local improvement strategy and the combination method in specific, and reported that those techniques could outperform previous approaches. In [4], both GA and Genetic Programming (GP) were examined on three different binary error correcting codes problems to generate optimal sets of codes, and devised a new chromosome representation, claiming benefits in certain conditions.

Power efficient design was addressed in many places. For example, a GA with integrated symbiotic mechanism was proposed in [5], to locate codes that provide single error correction and double error detection. The work formulated the selection of the parity check matrix into a collection of individual and specialized optimization problems. Another approach in [6] targeted the power consumption reduction in single error correcting, double error detecting checker circuits in memory, using the degrees of freedom in selecting the parity check matrix of the error correcting code, by using Simulated Annealing (SA) and GA to solve non-linear power optimization problems. Tests on actual memory traces of benchmarks indicated that considering power along with area and delay when selecting the parity check matrix could result in significant power reductions. Closely related work was reported in [7], which investigated the use of different evolutionary algorithms to improve the lower bounds for given parameters by relating this problem to the well-known Maximum Clique Problem. As in most of the evolutionary methods, local search mechanisms were integrated, like the one in [8], which presented a new local search algorithm for the error correcting codes problem called the Repulsion Algorithm (RA), and used it with a parallel GA to solve the problem, and compared it against a pure parallel GA. They achieved important improvement with the inclusion of the RA. In a related context, the authors of [9] resolved the question of the utility of the crossover operator in earlier studies on optimizing DNA error correcting codes, where the crossover operator in question was found to be substantially counterproductive and the majority of the crossover events produced results that violated the minimum distance constraints required for error correction.

Recently, few new evolutionary paradigms emerged, most importantly the one proposed in [10], called Chemical Reaction Optimization (CRO), to solve optimization problems by mimicking the interactions of molecules in a chemical reaction to reach a low energy stable state. The performance of the proposed algorithm was tested using three nondeterministic polynomial time hard combinatorial optimization problems; two traditional benchmark problems and a real-world problem, reporting competitive results compared with the existing successful metaheuristics. The authors employed this technique in [11] to the population transition problem, to maximize the probability of universal streaming by manipulating population transition probability matrix, and reported better performance than many commonly used methods for controlling population transition in many practical live streaming systems. Researchers have applied this technique to many intractable problems in various fields, like solving the grid scheduling problem in [12], which compared it with four generally acknowledged methods, and reported superior performance, and scheduling Directed Acyclic Graph (DAG) jobs in heterogeneous computing systems, and a Double Molecular Structure-based CRO (DMSCRO) method as in [13], to encode the execution order of the tasks in a DAG job, and the task-to-computing-node mapping, along with four elementary operations, and a fitness function suitable for DAG scheduling, and verified the effectiveness over a large set of randomly generated and real-world problems graphs, and testing the performance of CRO on three nondeterministic polynomial time hard combinatorial optimization problems reported in [10], claiming that it was very competitive with the existing metaheuristic, and outperformed them in some cases, like the real-world problem.

Among the other applications, [14] used it in developing an allocation algorithm to study three utility functions for utilization and fairness with hardware constraints, and showed that it outperformed the others by a good margin, and [15] used the CRO in allocating sets of maximally distant codes for a certain set of parameters, and reported good results in a relatively short time for small instances, and some difficulties in larger instances due to exploration reactions inefficiency, while [16] proposed an Adaptive CRO (ACRO) to alleviate the effort in tuning parameters, reducing the number of optimization parameters in canonical CRO along with an adaptive scheme to evolve them. They performed simulations on a widely-used benchmark of continuous problems claiming superior performance over canonical CRO. A multiobjective variant of CRO was reported in [17], called nondominated sorting CRO, and meant to exploit chemical reaction optimization features in tackling problems involving multiple criteria, making the multiobjective algorithm efficient from a computational cost viewpoint. Benchmarks test against reported good convergence.

Some novel approaches built on the CRO, for example the one in [18] proposed for training higher order neural networks, with two modifications; fixing the population size and using greedy reversible action after the regular actions. Compared to the basic CRO algorithm and two variants, using well known neural networks benchmarks, results reported significant statistical improvements. Another trend was to use hybrids. For example, [19] proposed a CRO with greedy strategy algorithm (CROG) to solve the binary Knapsack problem, by integrating a greedy strategy and random selection to repair the infeasible

solutions. The experimental results have shown superior performance compared to the standard GA, the Ant Colony Optimization (ACO), and the Quantum Inspired Evolution (QIE). The Orthogonal CRO (OCRO) is yet another example of hybridization reported in [2], adding quantization orthogonal crossover for global search. Tests on a set of several benchmark functions reported faster convergence speed to close to optimal solutions, especially for high-dimensional functions. Another hybrid was built on a local search to solve the Travelling Salesman problem (TSP) as in [21], by integrating the Lin-Kernighan (LK) local search, claiming better tradeoff between the exploration abilities of CRO and the exploitation abilities of LK local searcher, resulting in more efficient algorithm. The hybrid reported in [22], called Hybrid CRO (HCRO), was developed to solve task scheduling problems, by integrating a selection strategy with standard CRO. Both simulation and real-life experiments proved that the HCRO algorithm task scheduling was much better than the existing algorithms in terms of makespan and speed of convergence. Another hybrid was proposed in [23], by combining the two local search operators in the CRO with global search ability for global optimum, incorporating concepts from the CRO and the Particle Swarm Optimization (PSO). Tests on a set of twenty three benchmark functions have shown that this CRO and PSO hybrid could outperform the CRO in most of the experiments. Moreover, a novel computational method was reported in [24] as more robust and with less parameters than that used in the literature, called the Artificial Chemical Reaction Optimization Algorithm (ACROA), and applied it to multiple-sequence alignment and data mining.

Like any other evolutionary algorithm, parameters setting is one of major issues in the CRO implementation, and a study of their dynamics was reported in [25], covering various parameters. Numerical experiments for two test functions in the category of non-linear constrained optimization problems reported in the literature are carried out, indicating at par performance compared with other optimization methods.

Clearly, the CRO has a potential as an optimization method with universal applicability, but like all other algorithms, its parameters selection may lead to improper convergence sometimes. According to [26], when averaged over all the involved objective functions, all search algorithms behave exactly the same in terms of performance. All these algorithms are successful in solving different kinds of optimization problems. Hence an algorithm, compared to others may show equal performance on the average, but could outperform many others when matched to the right problem type.

III. Maximally Distant Codes

Data transmission and retention reliability requires the use of codes with error tolerance capability. Finding such sets of codes using exact search strategies requires hefty computational power even for a moderately sized instance. The objective of the search may vary; from finding a set of n-bit maximally distant codewords with predefined set size, to finding the largest set of n-bit codewords with some predefined minimum hamming distance. There can be many ways to map this problem to evolutionary algorithms, like a matrix representing a set of codewords as a candidate solution, or a binary vector representing ***m*** codes ***n*** bits each. Both of those mappings require a good deal of time to validate the solutions after reactions. A good mapping must provide a balanced memory and processing cost during the search to achieve acceptable utilization of resources. The maximally distant discovery problem maps well to the binary Knapsack problem with minimal validity test cost, and will be used in the two implementations; the standard and proposed version with enhanced exploration.

Also critical to the processing cost, and hence the search quality, is the ability of the mapping and the processing to offer diversification while producing valid solutions to avoid wasting computational resources. The maximally distant codes allocation problem lends itself easily to the binary Knapsack problem, where the ones and zeros in a vector representing a candidate solution indicate inclusion and exclusion of codewords of the corresponding positions. The problem is well suited to evolutionary search paradigms in general.

The theoretical bounds of ***A(n, d)*** stated in Table I are practically lose, and over a thousand papers have been written describing methods to improve those bounds, and the results till recent dates are partially shown in Table II as stated in [27].

TABLE II. Codewords Versus Minimal Distance Lower Bounds

	Hamming Distance		
n	***d = 3***	***d = 5***	***d = 7***
8	20	4	2
10	72	12	2
12	256	32	4
14	1024	128	16
16	2720 – 3276	256 – 340	36 – 37
18	10496 – 13104	1024 – 1280	128 – 142
20	36864 – 43688	2560 – 4096	512

IV. Chemical Reaction Optimization

The CRO algorithm starts with an initial set of randomly selected molecules, and iteratively applies one of four reactions until some stopping criteria is met. The exploitation reactions have an equal number of inputs and outputs and hence the population size remains fixed regardless of how often they are applied. On the contrary, the exploration reactions have no balance; they either decrease or increase the population size as they generate one out of two or two out of one, and unless they are equal in frequency, the population size grows to an unwanted limit or diminishes to one. Extremely large population size is a computational burden, while extremely small population size is inefficient in space exploration. Balancing the population size is important for a fruitful search.

To allocate sets of maximally distant codes, we need to maximize the minimum distance between any two codes in the set or the mean distance among all codes. Since the CRO is a

minimization strategy, codewords similarities will be used as a cost function for guiding the search. For ***n***-bit codewords, the similarity is in the range ***0*** to ***n-1***, and for **m** codewords, there exist ***m (m-1)/2*** values of similarity related to all pairs. Using a composite cost function with a balancing factor $0 \leq \mu \leq 1$, we used the following cost function:

$$C = \frac{2\mu}{m(m-1)} \sum_{i=1}^{m} \sum_{j=i+1}^{m} S_{ij} + (1-\mu) Max(Max(S_{ij})) \quad (2)$$

Where:

S is the Similarity matrix of m^2 entries.

m is the number of Codewords.

The CRO design has two major components, which are molecules that represent solutions, and elementary reactions, necessary to traverse the solution space looking for an optimal or near optimal solution.

A. Molecules

Molecules represent possible solutions to the problem and their characteristics make one molecule distinguishable from another. Chemically, bonds form and break acquiring and releasing energy, respectively. This energy exchange with the surroundings abides by the first law of thermodynamics, which states that energy can neither be created nor destroyed. The major attributes that describe a molecule are:

- Structure, which represents the solution currently held by a molecule. In our case, the molecule has the form of a binary vector whose size is equal to 2^n for ***n***-bit codewords.
- Potential Energy, which represents the energy corresponding to the structure, and it depicts the value of the objective function of the current molecular structure.
- Kinetic Energy, which represents the level of tolerance of a molecule to change to a less favorable structure, i.e., with higher potential energy.

B. Elementary Reactions

Elementary reactions are the means to explore various parts of the solution space and to exploit the most feasible neighborhoods. Reactions allow molecules to collide with each other or with the walls causing structural changes, making possible a product formation. There are two types.

- Exploitation

Exploitation is a process that tends to search for better solutions in the neighborhood of one. This is achieved through molecule collisions, either with the wall or with another molecule. Typically, molecule(s) go through subtle change hoping for better ones. Exploitation is carried out through two major reactions:

∗ *Wall Collision*

This is a unimolecular reaction in the form of a collision with a wall to bounce with some energy loss and a subtle structural deformation. The lost energy is stored in a central energy buffer and its amount is proportional to the sum of its kinetic energy and its potential energy gain. The factor used is called kinetic energy loss rate. This reaction performs local search with ability of escaping local minima.

∗ *Molecular Collision*

This is a bimolecular reaction, and it occurs when two molecules collide and bounce back, it is similar to that of wall collision in that these reactions are also not vigorous. The result of this reaction is a subtle structural change of the reactants.

- Exploration

Exploration is a process that tends to search for better subspaces for later exploitation, and hence escaping local optima. This is achieved through major structural change to molecules which are not capable of getting better anymore. Exploration is carried out through two major reactions, decomposition and synthesis, when molecules representing solutions exhaust chances of exploitation. Description and conditions of those reactions are detailed below.

∗ *Decomposition*

This is a unimolecular reaction, but it differs from the wall collision in that it produces two molecules instead of one, with relatively significant structural change. When a molecule hits a wall, it breaks up into two molecules with new structures. Typically, the new structures reflect major transformation, and hence it is important for exploration when the local search in a certain neighborhood is not feasible anymore. Energy is transferred from the buffer to sustain its change to form the new molecules.

∗ *Synthesis*

This is a bimolecular reaction, and it also occurs when two molecules collide and combine to form a single molecule. Synthesis reactants involve a vigorous change in their molecular structures, which is also necessary to reach out for new subspaces.

The following sections detail the four elementary reactions using a simple example to find 8 maximally distant 5-bit codewords, The reactants are shown in red color while the products are in blue color.

1) Unimolecular Reactions

There are two unimolecular reactions; wall collision and decomposition. In wall collision, a molecule hits the wall and bounces back with some deformation; a minor structural change.

One way to carry out this is to pick an entry at random and scan from there up, with wrap around, looking for an opposite one, to flip both. Another way to do it is to select two random numbers in the range 0 to 31 to index the entries to flip under the condition that the selected entries are opposite, i.e. one of them is 1 and the other is 0. Fig. 1 shows an example of wall collision reaction using two random cuts.

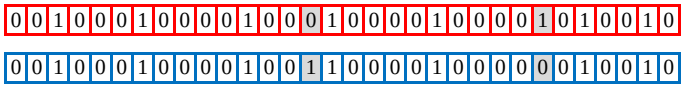

Fig. 1. Wall Collision Reaction

The second unimolecular reaction is decomposition. In this case, a molecule breaks up to make two new molecules with major structural differences from the original ones. Typically, two copies of the molecule are modified by selecting a random number in the range 0 to 31 to make a cut in the two structures, then the upper part of one and the lower part of the other are shuffled, by circulating those strings a random number of bits. Fig. 2 shows an example, the gray cells are the fixed parts, and the rest of each is the result of circulation.

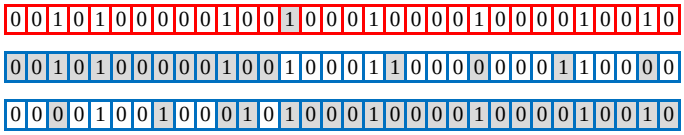

Fig. 2. Decomposition Reaction

2) Bimolecular Reactions

There are two reactions that involve two molecules as reactants. The first is synthesis, in which the two selected molecules are merged into one. A molecule is formed as a bitwise logical ***xor*** of the two molecules, then 1's or 0's are inverted at random to keep the numbers of 1's right; ***m*** entries. Fig. 3 shows an example. The gray cells are 1's, inverted 0's, to make the number of 1's equal 8 for a valid solution.

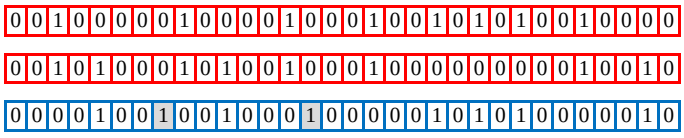

Fig. 3. Synthesis Reaction

The second bimolecular reaction is the molecular collision, where two molecules collide to bounce with subtle structural difference. In this case, a good deal of the properties of both are handed over to the resulting molecules. A random number in the range 0 to 31, is used to make cuts in the two reactants, then the upper parts are swapped, and if the solutions are invalid, having more or less 1's than ***m***, the molecules are scanned to invert 1's or 0's properly. Fig. 4 shows an example, the gray cell represents a random cut for both reactants, the upper parts are then swapped and the molecules are scanned to invert 1's or 0's at random to keep the numbers of 1's right; ***m*** entries. The gray cells in the products refer to inverted entries.

Iterative application of reactions to molecules representing solutions generates better ones converging to an optimal or near optimal solution. It is quite important for convergence to carry out transformations that provide balanced exploration and exploitation.

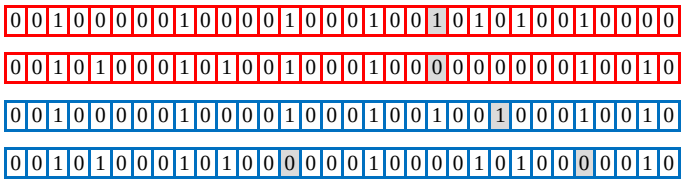

Fig. 4. Molecular Collision Reaction

Table III shows the significance of those reactions to the exploration and exploitation aspects.

The algorithm starts by setting initial parameters; population size as an initial number of molecules in the pool, molecular collision rate, kinetic energy loss rate, initial kinetic energy, buffer size, and decomposition and synthesis thresholds. Then, a set of initial molecules are generated at random, or through some vision for a more feasible initial set. A series of reactions, typically one at a time, are applied iteratively until the stopping criterion is satisfied, concluding with best possible solutions.

In each iteration, either one unimolecular or one bimolecular reaction is triggered, based on a preset rate called MoleColl.

TABLE III. ACTIONS AND SIGNIFICANCE

	Reactions: Types and Contributions		
Reaction	***Type***	***Exploration***	***Exploitation***
Wall Collision	Unimolecular	Minor	Major
Decomposition	Unimolecular	Major	Minor
Synthesis	Bimolecular	Major	Minor
Molecular Collision	Bimolecular	Minor	Major

V. BALANCED EXPLOITATION AND EXPLORATION

The exploitation and exploration balance of the standard implementation suffers most of the time, especially when used to solve large instances as reported by many researchers, resulting in low improvement rates after few thousands computations. It is quite hard to figure out how the search traverses the solution landscape, but improving the exploration reactions performance is necessary for convergence.

New search operators were found useful as stated in the literature. In this work, the performance was enhanced by relaxing the exploration reactions.

In the unimolecular path, decomposition reaction in the standard implementation takes place if the selected molecule satisfies the decomposition criterion, i.e., (number of hits − minimum hit number) > α, where α can be interpreted as the tolerance of duration for the molecule without obtaining any new local minimum solution. If so, the molecule will experience decomposition, otherwise it will go through a wall collision.

In the bimolecular path, synthesis reaction in the standard implementation takes place if the selected molecules, two or more, satisfy the synthesis criterion; the total kinetic energy is less than some preset minimum β. If it is satisfied by all the selected molecules, they combine through synthesis. Otherwise, they experience an molecular collision.

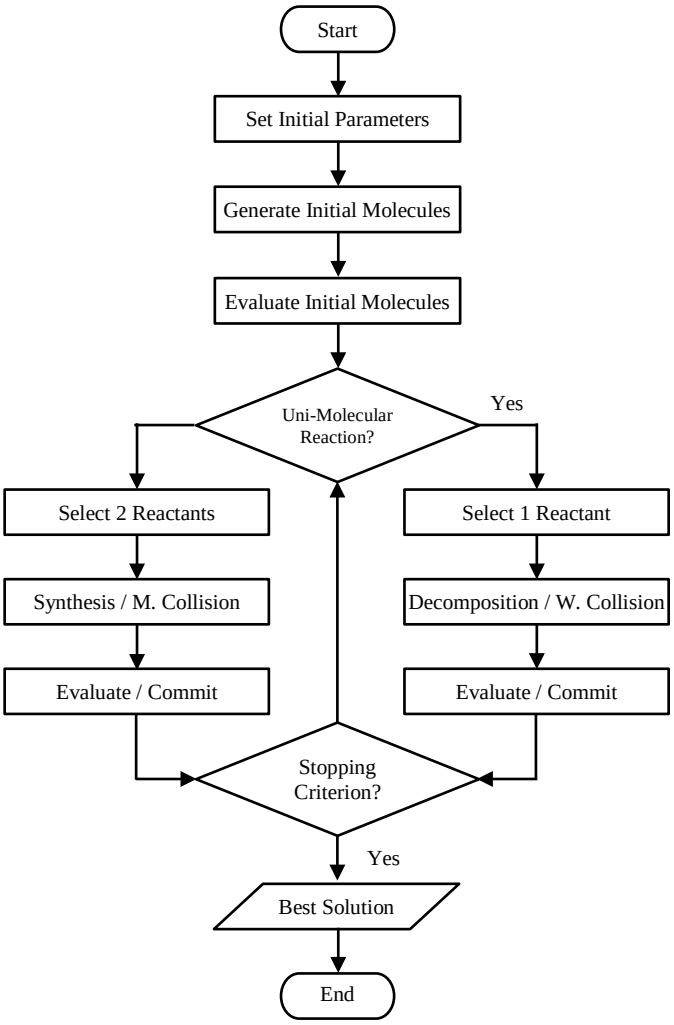

Fig. 5. Enhanced CRO Algorithm

The solutions generation function performs one of the four reactions, based on a preset rate MoleColl, shown in Fig. 5 as unimolecular or bimolecular decision. Otherwise, it performs unimolecular reactions. In the standard implementation, one of the two conditions DEC and SYN, standing for decomposition and synthesis discussed earlier, decide whether to carry out decomposition or synthesis, respectively, or one of the other reactions, wall collision or molecular collision, respectively, as shown in Fig. 6 and Fig. 7. The solution evolution is achieved by repetitively provoking one of the four elementary reactions. Convergence requires proper exploration, and triggering the decomposition and synthesis processes based on some static threshold may not be efficient.

The proposed enhancement has a different approach to trigger those processes. In the unimolecular path, the selected molecule is allowed to perform both decomposition and wall collision, but only one of them is committed and the other is abandoned. In the bimolecular path, the two selected molecules are allowed to perform both synthesis and molecular collision, and again only one of them is committed and the other is abandoned. Fig. 8 and Fig. 9 shows pseudocode for the enhanced reaction triggers.

```
Procedure UnimolecularReaction
If DEC  {
        ApplyDecompositionReaction
        }
    Else
        {
        ApplyWallCollisionReaction
        }
End procedure
```

Fig. 6. Standard Unimolecular Reactions

```
Procedure BimolecularReaction
If SYN  {
        ApplySynthesisReaction
        }
    Else
        {
        ApplyMolecularCollisionReaction
        }
End procedure
```

Fig. 7. Standard Bimolecular Reactions

```
Procedure UnimolecualarReaction
        MakeDecompositionReaction
        MakeWallCollisionReaction
        SelectReactionRoulette
        CommitReaction
End procedure
```

Fig. 8. Enhanced Unimolecular Reactions

```
Procedure BimolecualarReaction
        MakeSynthesisReaction
        MakeMolecularCollisionReaction
        SelectReactionRoulette
        CommitReaction
End procedure
```

Fig. 9. Enhanced Bimolecular Reactions

In this reaction triggering scheme, the exploration reactions compete with exploitation reactions on two to one basis in the unimolecular types and on one to two bases in the bimolecular types. This means favoring the exploitation in the bimolecular reactions and favoring the exploration in the unimolecular reactions. The process is randomized but the fitness of the products plays a role in the triggering process. This scheme may lead to undesired increase in the population size, due to favoring the decomposition over synthesis, which can be resolved by a check against lower and upper bounds, to reinsert and drop molecules based on some criterion like the fitness value.

VI. EXPERIMENTS

In the initialization phase, PopSize, KELossRate, MoleColl, Buffer, InitialKE, α, and β were configured based on the literature recommendations, and the initial set of molecules is generated at random. Their initial potential energies are determined by their corresponding objective function values while their initial kinetic energies were set to InitialKE.

Table IV shows the standard algorithm performance reported earlier. The target was to locate sets of maximally distant codes of 8-bit strings, for three cost calculation scenarios: maximum similarity (corresponding to minimum distance), mean similarity (corresponding to mean distance) distance, and the two together with equal significance.

Table V shows the enhanced exploration method impact, using the same instances and initial population reported earlier. Although marginal, test results demonstrate better performance especially using the minimum distance metric.

TABLE IV. IMPACT OF THE WEIGHTING PARAMETER ON SEARCH

	Weighting *Parameter*					
	μ = 0.0		***μ = 0.5***		***μ = 1.0***	
Set Size	***Mean***	***Min***	***Mean***	***Min***	***Mean***	***Min***
8	4.43	4	4.46	4	4.57	3
16	4.06	2	4.27	2	4.27	2
32	3.77	1	4.13	1	4.13	1
64	3.72	1	3.86	1	3.86	1

TABLE V. IMPACT OF THE WEIGHTING PARAMETER ON SEARCH

	Weighting *Parameter*					
	μ = 0.0		***μ = 0.5***		***μ = 1.0***	
Set Size	***Mean***	***Min***	***Mean***	***Min***	***Mean***	***Min***
8	4.52	4	4.57	4	4.68	3
16	4.29	3	4.36	2	4.39	2
32	3.95	2	4.21	1	4.22	1
64	3.88	1	3.93	1	3.96	1

Table VI shows the performance of the algorithm in locating sets of 10-bit codewords with various set sizes. The proposed enhancement outperformed the standard, in both minimum distance and mean distance metrics. Many of those runs were repeated many times with the same initialization for fair comparison.

Table VII shows comparative performance of the two implementations in locating sets of 12-bit codewords with various set sizes. Table VIII shows comparative performance of the algorithms in finding three sets of fixed size and codeword length. Clearly the enhanced exploration is better in at least the mean distance metric if not in both.

TABLE VI. PERFORMANCE AGAINST SET SIZE (10-BIT CODEWORDS)

	Standard		***Enhanced***	
Set Size	***Mean***	***Min***	***Mean***	***Min***
8	5.34	4	5.68	5
16	5.13	3	5.26	4
32	5.09	3	5.12	4
64	5.08	2	5.07	3
128	5.08	2	5.08	2

TABLE VII. PERFORMANCE AGAINST SET SIZE (12-BIT CODEWORDS)

	Standard		***Enhanced***	
Set Size	***Mean***	***Min***	***Mean***	***Min***
32	6.42	4	6.68	4
64	6.26	3	6.42	4
128	6.15	3	6.28	3
256	6.12	3	6.14	3
512	6.10	2	6.11	2

TABLE VIII. PERFORMANCE AGAINST CODEWORD LENGTH / SET SIZE

	Standard		***Enhanced***	
Code Length / Set Size	***Mean***	***Min***	***Mean***	***Min***
10-bit / 16	5.11	4	5.36	5
12-bit / 64	6.13	3	6.41	4
14-bit / 256	7.24	3	7.58	3

TABLE IX. SUCCESS RATE COMPARISON

			Success Rate	
Code	***Bound***	***Computations***	***Standard***	***Enhanced***
8, 16, 3	28	16,000	12	14
8, 4, 5	6	4,000	11	13
10, 64, 3	93	64,000	11	14
10, 16, 5	18	16,000	10	11
12, 256, 3	315	256,000	11	13
12, 32, 5	51	32,000	9	14
14, 1024, 3	1092	1,024,000	8	10
14, 128, 3	154	128,000	8	9

The power of finding solutions is expressed as success rate as shown in Table IX, where the two implementations were run 20 times each with the objective of finding sets of 4 code lengths: 8, 10, 12 and 14, and two variations of each.

The number of computations was based on the space size. The proposed enhanced exploration implementation demonstrated a marginal improvement over the standard, in terms of the number of times it could locate a solution.

The advantage of the enhanced exploration is shown in Fig. 10 as progress plot, mean distance and minimum distance of the best solution over time. The standard implementation stabilized within few thousands of iterations, while the enhanced kept improving for longer time, indicating better exploration power. The enhanced strategy resulted in larger fluctuations of the mean distance over time due to the continuous generation of solutions in new subspaces, and hence better exploration with increased time budgets.

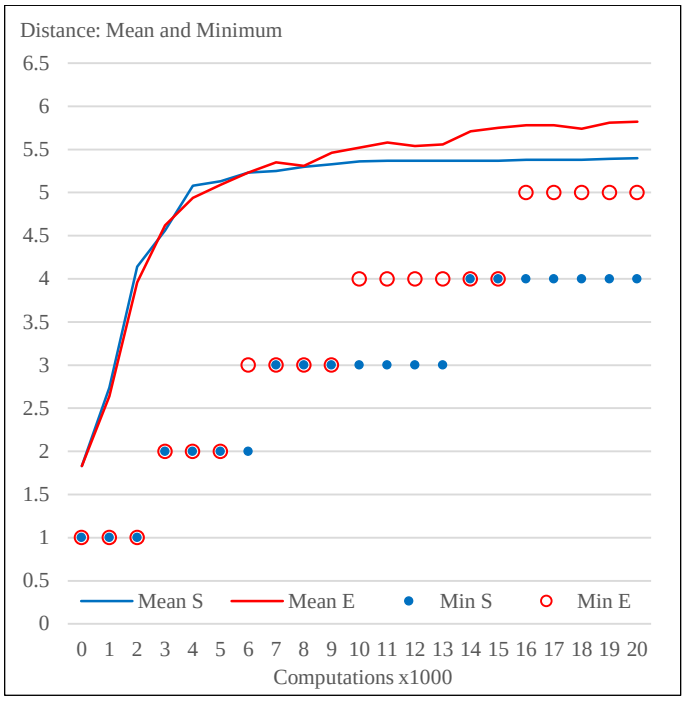

Fig. 10. Search Progress; Locating 12-bit Codeword Sets

Tests were run on i7 Intel quad core processor based desktops with 16 GB DRAM. Typical runs took from few minutes for small instances to few tens of minutes for the largest, carrying out tens of thousands computations, until improvement rate reaches 0.1%.

VII. Conclusion

Solving the Mapping the Maximally Distant Codes Allocation Problem by the Chemical Reaction Optimization algorithm has been reported in literature with some problems especially with larger instances. The issue is that the exploration reactions consume a good deal of fruitless computations. In this enhancement, we proposed a different approach to manage the exploration reactions. Instead of using preset threshold based triggering, causing unpredicted reactions rate, we involved the fitness of the selected molecules and randomization process to trigger those reactions. Using various metrics, progress or fitness change over time and success rate, the proposed triggering methods outperformed the standard implementation.

Acknowledgment

This research was conducted during the academic year 2014/2015, supported by Jordan University of Science and Technology in Irbid - Jordan, as a sabbatical leave at New Jersey Institute of technology in Newark, New Jersey USA.

References

[1] K. Dontas, K. De Jong, "Discovery of maximal distance codes using genetic algorithms," Proceedings of the 2nd International IEEE Conference on Tools for Artificial Intellegence, Herndon, Verginia, USA, pp. 805-811, November 1990.

[2] W. Haas S. Houghten, "A comparison of evolutionary algorithms for finding optimal error correcting codes." Proceedings of the 3rd IASTED International Conference on Computational Intelligence. Anaheim, CA, pp. 64-70, July 2007.

[3] C. Cotta, "Scatter search and memetic approaches to the error correcting code problem," Evolutionary Computation in Combinatorial Optimization, 4th European Conference, Coimbra, Portugal, pp. 51-61, April 2004.

[4] D. McCarney, S. Houghten and B. Ross, "Evolutionary approaches to the generation of optimal error correcting codes," Proceeding of the 14th annual conference on genetic and evolutionary computation, ACM, New York, USA, pp. 1135-1142, 2012.

[5] H. Lee and E. Kim, "A symbiotic evolutionary design of error correcting code with minimal power consumption," Electronics and Telecommunications Research Institute Journal, Koreya, vol. 30(6), pp. 799-806. 2008.

[6] S. Ghosh, S. Basu and N. Touba, "Selecting error correcting codes to minimize power in memory checker circuits," Journal of Low Power Electronics, vol. 1(10), pp. 63-72. 2005.

[7] W. Hwanga, C. Oua, C. Hsub, T. Loc, "Iterative optimization for joint design of source and channel codes using genetic algorithms," Journal of the Chinese Institute of Engineers, 2005, vol. 28(5), pp. 803-810.

[8] E. Alba, F. Chicano. "Solving the error correcting code problem with parallel hybrid heuristics," Proceedings of the 2004 ACM research Symposium on Applied Computing. 2004, pp. 985-989.

[9] J. Bland, " Local search optimization applied to the minimum distance problem," Journal of Advanced Engineering Informatics archive, Elsevier Science Publishers, 2007. vol. 21(4), pp. 391-397.

[10] A. Lam and V. Li. "Chemical reaction inspired metaheuristic for optimization," IEEE Transactions on Evolutionary Computation, vol. 14(3), pp. 381-399, 2010

[11] A. Lam, J. Xu and V. Li, "Chemical reaction optimization for population transition in peer-to-peer live streaming," IEEE World Congress on Computational Intelligence, Barcelona, Spain, pp. 1-8, July 2010.

[12] J. Xu, A. Lam and V. Li, "Chemical reaction optimization for the grid scheduling problem," 2010 IEEE Conference on Communications, Cape Town, pp. 1-5, May 2010.

[13] Y. Xu, K. Li, L. He and T. Truong, "A directed acyclic graph scheduling scheme on heterogeneous computing systems using double molecular structure based chemical reaction optimization," Journal of Parallel and Distributed Computing, Academic Press, Inc, Orlando, Florida, USA, pp. 1306-1322, vol. 73(9), September 2013.

[14] A. Lam and V. Li. "Chemical reaction optimization for cognitive radio spectrum allocation," IEEE Communications Society, Proceedings of Globecom, Miami, Florida, USA, pp. 1-5, December 2010.

[15] T. Eldos, W. Nazih and H. Elsimary, "Error correction code allocation using the chemical reaction optimization algorithm," International Journal of Engineering and Computer Science, vol. 13(3), pp. 54-59, 2013.

[16] J. Yu, A. Lam and V. Li, "Adaptive chemical reaction optimization for global numerical optimization," Neural and Evolutionary Computing, Cornell University Library, Computer Science, July 2015, (http://arxiv.org/abs/1507.02492v1).

[17] S. Bechikh, A. Chaabani, L. Ben Said, "An efficient chemical reaction optimization algorithm for multiobjective optimization," IEEE Transactions on Cybernetics, vol. 45(10), pp. 2051-2064, 2014.

[18] K. Sahu, S. Panigrahi and H. Beehera, “A novel chemical reaction optimization algorithm for higher order neural network training,” Journal of Theoretical and Applied Information Technology, vol. 53(3), pp. 402-409, July 2913.

[19] T. Truonga, K. Lia and Y. Xua, "Chemical reaction optimization with greedy strategy for the 0–1 knapsack problem," Applied Soft Computing,Elsiver, vol. 13(4), pp. 1774-1780, April 2013.

[20] Z. Li, T. Nguyen and S. Chen, “Orthogonal chemical reaction optimization algorithm for global numerical optimization problems,” Expert Systems with Applications, vol. 42 (6), pp. 3242-3252, April 2015.

[21] J. Sun, Y. Wang, J. Li and K. Gao, “Hybrid algorithm based on chemical reaction optimization and Lin-Kernighan local search for the traveling salesman problem, Shanghai,” Seventh International Conference on Natural Computation, vol. (3), pp. 1518-1521, July 2011.

[22] Y. Xu, K. Li, L. He, L. Zhang and K. Li, "A hybrid chemical reaction optimization scheme for task scheduling on heterogenous computing sysems," IEEE transactions on Parallel and Distributed Systems, vol. 26(12), pp. 3208-3222, December 2014.

[23] T. Nguyen, Z. Li, S. Zhang and T. Truong, “A hybrid algorithm based on particle swarm and chemical reaction optimization,” Expert Systems with Applications, vol. 41(5), pp. 2134-2143, April 2014.

[24] Bilal Alatas, "ACROA: Artificial Chemical Reaction Optimization Algorithm for global optimization," Expert Systems with Applications: An International Journal, vol. 38(10), pp. 13170-13180, September 2011.

[25] S. Pandharipande and A. Kumar, “Comparative study of performance of chemical reaction optimization with genetic algorithm,” International Journal of Computer Applications, vol. 107(8), pp. 1-8, December 2014.

[26] D. Wolpert and W. Macready, “No free lunch theorems for optimization,” IEEE Transactions on Evolutionary Algorithms, vol. 1(1), pp. 67-82, 2002.

[27] Henry S. Warren, "Hacker's Delight," Addison-Wesley, pp. 350-351, September 25, 2012.