Journal of Computer Science

Original Research Paper

Maximally Distant Codes Allocation Using Chemical Reaction Optimization and Ant Colony Optimization Algorithms

[1,3]**Taisir Eldos,** [2]**Waleed Nazih and** [1]**Aws Kanan**

[1]*Department of Computer Engineering, Prince Sattam Bin Abdulaziz University, Alkharj, Saudi Arabia*
[2]*Department of Computer Science, Prince Sattam Bin Abdulaziz University, Alkharj, Saudi Arabia*
[3]*Department of Computer Engineering, Jordan University of Science and Technology, Irbid, Jordan*

Article history
Received: 10-05-2015
Revised: 11-09-2015
Accepted: 17-10-2015

Corresponding Author:
Taisir Eldos
Department of Computer Engineering, Prince Sattam Bin Abdulaziz University, Alkharj, Saudi Arabia;
Department of Computer Engineering, Jordan University of Science and Technology, Irbid, Jordan
Email: eldos@sau.edu.sa; eldos@just.edu.jo

Abstract: Error correcting codes, also known as error controlling codes, are set of codes with redundancy that allows detecting errors. This is quite useful in transmitting data over a noisy channel or when retrieving data from a storage with possible physical defects. The idea is to use a set of code words that are maximally distant from each other, hence reducing the chance of changing a valid codeword to another valid codeword by flipping bits. The problem can be viewed as picking m codes out of 2^n available n-bit combinations, such that the aggregate hamming distance among those codewords is maximized. Due to the large solution spaces of such problems, greedy algorithms are sometimes used to generate quick and dirty solutions. However, modern evolutionary search algorithms like genetic algorithms, swarm particles, gravitational search and others, offer good alternatives, yielding near optimal solutions in exchange for some time. Chemical Reaction Optimization (CRO) has emerged as a new evolutionary algorithm to solve complex optimization problems. This algorithm mimics the molecular interactions towards finding a minimal energy state, which corresponds to an optimal solution for the problem in hand. In this research, we proposed a solution for the maximally distant codes allocation problem, through a binary knapsack mapping and compared the performance with the well established Ant Colony Optimization (ACO) algorithm, which is inspired by the ant's capability to find the shortest path between the nest and source of food. The binary knapsack mapping was used in the two algorithms. Test results showed that the CRO outperformed the ACO in every metric given any time budget.

Keywords: Maximally Distant Codes, Evolutionary Algorithms, Chemical Reaction Optimization, Ant Colony Optimization

Introduction

Allocating sets of codes with maximum aggregate mutual distances for use as error control codes is of great significance and finding optimal solutions for practically sized problems using full search is a challenge due to the prohibitively large solution spaces. For example, the solution space of the small instance (7, 16, 3), which requires finding a set of 16 codewords of 7 bits with minimal mutual Hamming distance of 3, is at least 10^{20}, ruling out any exact search methodology. Table 1 shows various code lengths n and the number of codewords with minimal distance d of 3 and 5.

Evolutionary optimization algorithms offer optimal or near optimal solutions in reasonable time. Many evolutionary algorithms have been used to solve complex problems with varying time and quality tradeoffs. The CRO and ACO algorithms have emerged recently as new methods to efficiently explore such large spaces with reasonable computational resources. In this study, we mapped the maximally distant code allocation problem to the well known binary knapsack problem and compared the performance of those two algorithms in finding sets of codewords of various length and cardinality, using a weighted fitness (or cost) function to provide balance between two objectives; mean and minimum distance.

Table 1. Codewords with minimal distance

n	d=3	d=5
6	8	2
8	20	4
10	72-79	12
12	256	32
14	1024	128
16	2560-3276	256-340

Literature Review

Dorigo and Gambardella (1997), proposed an early work of ACO; an artificial ant capable of solving both symmetric and asymmetric Traveling Salesman Problem (TSP), using a natural metaphor to design an optimization algorithm, through a sensibility analysis to help tune the parameters of an ant colony model where ants leave a fuzzy trace of pheromone to mark their track and its neighborhood. Recently, many complex problems were solved by variants of the ACO algorithm with great success; Baterina and Oppus (2010) presented an edge detection technique by establishing a pheromone matrix representing the edge information at each pixel based on the routes formed by the ants dispatched on the image. The ants movement was guided by the variation in the intensity values. Experimental results have shown the success of the technique in extracting edges from a digital image. Yu *et al*. (2005) developed a coarse-grain parallel ACO algorithm to develop an optimization model for bus transit network based on road network, aiming at achieving maximum passenger flow and minimum transfers per unit length with non-linear rate and line length as constraints. It used a heuristic pheromone distribution rule, by which path searching activities are adjusted according to the objective value. Hung *et al*. (2007) reported analysis using a lower pheromone trail bound and a dynamic updating rule for the heuristic parameters based on entropy to improve the efficiency of the algorithm in solving the TSP, with extremely large problem space and claimed superior search performance over traditional ACO algorithms. Lorpunmanee *et al*. (2007) addressed the scheduling problem by developing a general framework of grid scheduling using dynamic information and an ACO algorithm to improve making decisions, by comparing its performance with various dispatching rules such as First Come First Served (FCFS), Earliest Due Date (EDD), Earliest Release Date (ERD). G´omez (2005) proposed the Omicron ACO (OA); a population-based ACO alternative originally designed as an analytical tool and proved the advantages of the OA by experimentally comparing the behavior of the OA and the MMAS as a function of time. The ACO algorithm performance has been boosted by involving other techniques in its internal workings; for example, Zaferanieh *et al*. (2009) used an ACO and Simulated Annealing (SA) algorithms to find the core of a graph, such that the total travel cost time required for the demand points to reach the closest vertex on this path is minimized. Ginidi *et al*. (2010) developed a new fuzzy-logic based ACO algorithm, taking into consideration the uncertainties that can be found in both the heuristic and the pheromone factors, by considering fuzzy levels in calculating the involved parameters. They proposed a stochastic-based technique to enable the artificial ant to choose the best oncoming step based on the values of the probabilities and their corresponding fuzzy levels. The proposed algorithm gave the optimal solution in a form of an optimal value and its corresponding fuzzy level, using benchmark Quadratic Assignment Problem (QAP) and TSP. Ho *et al*. (2006) proposed an algorithm that incorporated key features of the tabu-search method in the development of a relatively simple but robust global ACO algorithm and used numerical results to validate and demonstrate the feasibility and effectiveness of the proposed algorithm in solving Electromagnetic (EM) design problems. Shieh *et al*. (2003) focused on the transmission of codebook indices in a noisy environment, to minimize the impact of channel noise, using ACO to find a suitable index assignment and reported that the channel distortion was substantially reduced without incurring extra cost such as that in error-detection code and error-correction code. Eldos *et al*. (2013a) used the ACO algorithm to solve the Printed Circuits Boards Drilling Problem (PCBDP), by finding the best order to drill each set of holes of the same diameter. Kanan *et al*. (2013) used the ACO in solving the routing problem in ad hoc mobile networks.

Haas and Houghten (2009) compared the performance of many evolutionary algorithms with local search and greedy methods, in solving the error-correcting-code 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. Bland (2007) 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.

Hwanga *et al*. (2005) investigated the use of different evolutionary algorithms for improving the lower bounds for given parameters by relating this problem to the well known Maximum Clique Problem. Lacan and Chatonnay (1999) presented an algorithm for the joint design of source and channel codes, where Channel-Optimized Vector Quantization (COVQ) and Rate-Punctured Convolutional Coding (RCPC) were used to design the source and channel codes, respectively. They used a Genetic Algorithm to allow the design of COVQ escape poor local optima and to reduce the time required for realizing the unequal error protection scheme best matched to the COVQ. Test results proved that their method was an effective alternative for cases with high rate-distortion performance and low computational resources.

Alba and Chicano (2004) have shown some promising results of computations as an answer to why and how the GA was used for this problem. Cotta (2004) tackled the error correcting codes allocation with two related techniques, Memetic algorithms and Scatter search and investigated the instantiation of those techniques for error correction codes design; the design of the local improvement strategy and the combination method in specific. Tests showed that these techniques could outperform previous approaches to this problem. Ghosh *et al.* (2005) proposed an approach to reduce the power consumption 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. They used Simulated Annealing and Genetic Algorithms to solve the non-linear power optimization problem. Tests on actual memory traces of Spec and MediaBench benchmarks indicated that considering power along with area and delay when selecting the parity check matrix could result in power reductions of up to 27 and 41% for Hsiao and Hamming codes, respectively.

Lee and Kim (2008) presented the Repulsion Algorithm, as a new local search algorithm for the problem using a hybrid between Parallel Genetic Algorithm and this new algorithm and compared it against a pure Parallel Genetic Algorithm. The results showed that an important improvement was achieved with the inclusion of the Repulsion. The genetic algorithm equipped with the symbiotic mechanism was used to design a power-efficient ECC which provided 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 and proposed a symbiotic evolution method to search for an ECC with minimal power consumption.

Lam *et al.* (2010) presented an optimization algorithm to solve the population transition problem, to maximize the probability of universal streaming by manipulating population transition probability matrix. They employed a metaheuristic inspired by the chemical reaction process and called it CRO, to solve the problem. Simulations showed that it outperformed many commonly used methods for controlling population transition in many practical P2P live streaming systems. Xu *et al.* (2010) proposed a CRO algorithm for the grid scheduling problem and compared it with four generally acknowledged methods and showed that it performed the best. Lam *et al.* (2010) developed an allocation algorithm based on the recently proposed CRO, to study three utility functions for utilization and fairness, with the consideration of the hardware constraint and showed that it always outperformed the others by a good margin. Lam and Li (2010a) tested the performance of CRO on three nondeterministic polynomial-time hard combinatorial optimization problems, a real-world problem and two traditional benchmark problems. Simulations showed that CRO was very competitive with the existing metaheuristic and outperformed them in some cases, like the real-world problem.

McCarney *et al.* (2012) examined both genetic algorithm and genetic programming on three different binary error correcting code problems to generate optimal sets of codes. They devised a new chromosome representation, claiming benefits in certain conditions. Eldos *et al.* (2013b) used the CRO in allocating sets of maximally distant codes for a certain set of parameters, to provide for error control and reported good results in a relatively short time.

As Wolpert and Macready (2002) stated in their no free lunch theorems for optimization, the ACO and CRO have equal performance as the others on average, but could outperform all other metaheuristic when matched to the right problem type.

Maximally Distant Codes

Maximally distant codes are of great significance in data transmission due to their error tolerance capability. The search spaces for these codes are so large and hence exhaustive search strategies are ruled out, even for small instances. The problem is to find an n-bit m-codeword set of maximum mean mutual distance or maximum minimum distance, or to find the largest set of n-bit codewords with minimum hamming distance. The problem can be viewed as a binary knapsack problem, where the 1's indices represent the codewords that belong to the required set. The problem is only complex in terms of the prohibitively large solution space. However, the evolutionary search paradigms work well on such problems and we are going to compare the performance of two such optimization algorithms, one is a maximizer and the other is a minimizer.

Based on the application, the objective can be a given number of codewords with maximum mutual distance, or largest number of codewords with a given minimal distance. In this study, our objective is to compare the CRO and ACO performance in allocating a set of codewords with minimal distance.

There are three measures to use to guide the search process. The CRO uses a cost function to minimize, while the ACO uses a fitness function to maximize. The cost is related to how similar the codewords are, while the fitness is the opposite; how distant the codewords are. For n-bit codewords, the maximum similarity is n-1, while the minimum distance is 1. Now, for m codewords, there exists m (m-1)/2 values (similarities or distances) related to those pairs and one can use the mean, the mean of maxima of all codewords, or the maximum overall.

In the early implementation, we used the two extremes; the maximum similarity and the mean similarity for the CRO and the minimum distance and mean distance for the ACO, through a balancing factor $0 \le \mu \le 1$. The cost and fitness functions for the CRO and ACO respectively were:

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

$$F = \frac{2\mu}{m(m-1)} \sum_{i=1}^{m} \sum_{j=i+1}^{m} D_{ij} + (1-\mu) Min(Min(D_{ij})) \tag{2}$$

Where:

S	=	The Similarity matrix of m^2 entries
D	=	The Distance matrix of m^2 entries
m	=	The number of Codewords

The performance was less than expected in terms of the minimum distance especially when the codeword set size is $m = 2^{n-3}$. So, we opted to use the mean of maxima and mean of minima of all codewords as cost and fitness functions for the CRO and ACO respectively, as shown in Equation 3 and 4. We compared the performance of the two algorithms using the two approaches in guiding the search while judging the quality of the solutions using the same metrics; the mean and minimum distance:

$$C = \frac{1}{m} \sum_{i=1}^{m} \sum_{j=1, j \ne i}^{m} Max(S_{ij}) \tag{3}$$

$$F = \frac{1}{m} \sum_{i=1}^{m} \sum_{j=1, j \ne i}^{m} Min(D_{ij}) \tag{4}$$

Chemical Reaction Optimization

The CRO algorithm starts with an initial set of randomly selected molecules and applies predefined actions until some stopping criteria are met. The exploitation actions 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 actions either decrease or increase the population as they generate one out of two or two out of one and unless they are equal in frequency, the population size may reach unwanted limits. Extremely large population is undesirable because it acts as computational burden and extremely small population is undesirable because it reduces the effectiveness in exploring the solution space. The algorithm should provide control over to keep it within reasonable limits.

CRO Elementary Actions

The elementary actions of the CRO are divided into four types, based on the input and output cardinality. The following sections detail the actions using an example of finding 8 maximally distant codewords of 5-bit.

A1T1, 1-to-1 action, or deformation; one molecule is involved to produce one molecule. In this process, the molecule is deformed through a minor or a major structural change. We 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. Example:

Input

0	0	1	0	1	0	0	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	0

Output

0	0	1	0	1	0	0	1	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	0

A1T2, 1-to-2 action, or decomposition; one molecule is involved to produce more than one molecule; typically two molecules are derived from one. We make a copy of the molecule and select a random number in the range 0 to 31 to make a cut and then we shuffle the upper part of the first and the lower part of the second. Shuffling can be carried out by circulating the string a certain number of bits at random or through any other scheme. In the example below, the isolated position represent the shuffling process, while the continuous ones are outcomes of the copying process:

Input

0	0	1	0	1	0	0	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	0

Output 1

0	0	1	0	1	0	0	0	0	0	1	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	1	1	0	0	0	0

Output 2

0	0	0	0	1	0	0	1	0	0	0	1	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	0

A2T1, 2-to-1 action, or combination; two or more molecules are merged into one molecule; a process in which the properties of two or more molecules are passed to a new one. We form a molecule with permanent 1's and 0's where the two input molecules have 1's and 0's respectively, then fill the rest at random by 1's to complete the set of codeword. In this example, the gray positions represent voted 1's and 0's, 4 positions are left to be filled at random by 3 missing 1's:

Input 1

0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	1	0	1	0	1	0	0	1	0	0	0	0

Input 2

0	0	1	0	1	0	0	0	1	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	0

Output

0	0	1	0	1	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	1	0	0	1	0	0	0	0

A2T2, 2-to-2 action, or collaboration; two molecules are involved to produce two molecules; a process in which properties of both are passed into two new molecules. We select two random numbers in the range 0 to 31, or use one for both and make cuts in the two molecules, then form two molecules; one from the lower part of the first and the upper part of the second and the second from the lower part of the second and the upper part of the first. This process may produce invalid solutions, more or less 1's than the required size of codewords set and hence the two new molecules are scanned to randomly insert 1's instead of some 0's or 0's instead of some 1's, all at random. In this example, a single random cut is selected for both, the lower part of input 1 is copied to the lower part of output 1 and the upper part of input 2 is copied to the upper part of output 1, the upper part is then scanned to flip some 0's into 1's or some 1's into 0's such that the total number of 1's is equal to the size of the codeword set. In this case, the isolated gray positions represent 0's converted to 1's:

Input 1

0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	1	0	1	0	1	0	0	1	0	0	0	0

Input 2

0	0	1	0	1	0	0	0	1	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	0

Output 1

0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	1	0	0	1	0

Output 2

1	0	1	0	0	1	0	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	0

In the context of transforming a set of solutions into a new one, hopefully of better quality, we name the first two actions, the ones that involve one molecule, D-type actions; Deformation and Decomposition, while we name the other two, the ones that involve two molecules, C-type actions; Combination and Collaboration. Most of the literature uses chemical reactions terminology like synthesis instead of combination and intermolecular collision instead of collaboration. We opted to use terms that express the processes behavior as they take place in the search process.

Successive application of those actions to sets of molecules representing solutions over and over generates better ones. It is quite important for convergence to carry out operations that provide both exploration and exploitation. Table 2 shows the significance of those actions to the exploration and exploitation.

Table 2. Actions and significance

Action	Exploration	Exploitation
A1T1 (Deformation)	minor	major
A1T2 (Decomposition)	major	minor
A2T1 (Combination)	major	minor
A2T2 (Collaboration)	minor	major

```
procedure CRO
        initialization()
        while(stopping condition false)
                solutionsGeneration()
                daemonActions()
                energyUpdate()
                solutionUpdate
        end while
end procedure
```

Fig. 1. Generic CRO algorithm

```
pick a random number p ϵ [0,1]
if p > car  {
            if dcs apply  A1T2 else apply  A1T1
            }
else
            {
            If ccs apply  A2T1 else apply  A2T2
            }
```

Fig. 2. CRO actions provoking mechanism

The CRO algorithm starts with an initial set of solutions (molecules) and an initial set of parameters based on the instance to solve and goes through a repetitive process of applying actions until a stopping criterion is satisfied. In each iteration, one of the C-type actions or D-type actions is carried out based on a parameter called the C-type actions rate that determines the percentage of the C-type actions.

In each path, whether it is C-type or D-type, a test is carried out to see if the combination condition or decomposition condition, respectively, is met, or else collaboration or deformation, respectively, are carried out instead. The pseudocode in Fig 1 outlines the CRO algorithm behavior.

The solutions generation function performs one of the four actions, based on a preset rate (car) which defines the c-type actions rate. Otherwise, it performs d-type actions. Within the c-type or d-type code, the DCS and CCS conditions decide whether to carry out decomposition and combination, respectively, or the other actions. The four elementary actions are provoked based on the pseudocode in Fig. 2.

Typically, we used the number of computations as stopping criteria, with varying values of car, CCS and DCS based on the objective of the test run.

Ant Colony Optimization

This algorithm is inspired by the ants' moves between nests and food sources. Starting with random walk looking for food, ants place pheromone on the route, which accumulates against evaporation, leaving more of it on shorter routes and less on the longer unused ones, as routes with stronger pheromone appeal more to the ants.

The problem of maximally distant codes allocation maps straightforwardly to the binary knapsack problem; 5-bit codes are 32 and using an array of 32 entries representing 1 or 0 for take and leave respectively, would make it easy for the ant, as an agent generating a solution, to decide on which codeword to take or leave by setting or resetting its corresponding index. The pheromone matrix in this case is 32x32 symmetric entries with zero diagonal. For each solution, the fitness function decides how much pheromone is added to each link between any two codewords in the solution. The idea is to place pheromone on edges connecting all codewords of a candidate solution in proportion to the fitness of that solution. Agents start solutions by selecting a random entry (corresponding to a codeword) and select the next entry based on the largest entry in the pheromone matrix. Now the two will have to decide on the third entry based on the sum (or average) of pheromone and the three of them decide on the fourth and so on. With a large number of bits n, the pheromone matrix becomes extremely large for processing, 2^{2n} entries. Instead, we opted to change for a model that is less complex with a little compromise, where the pheromone is accumulated on the nodes rather than edges, this way we need only a vector of 2^n entries. In the original model, an ant makes a random starting point then walks stochastically to the other nodes based on the edges pheromone and distance of the nodes. We used a simple approach that lets an ant decide on the next node based on group thinking; all the accumulated nodes decide who to join them next. After evaluating the fitness of the solutions, the pheromone of every entry is updated in proportion to the fitness of every solution it was taken as part of, relative to the maximum fitness so far. An outline of the algorithm is shown in Fig 3.

A set of ants generate a set of solutions whose fitness values are used to update the pheromone values.

```
procedure ACO
    initialization()
    while(stopping condition false)
        solutionsGeneration()
        daemonActions()
        pheromoneUpdate()
        solutionUpdate
    end while
end procedure
```

Fig. 3. Generic ACO algorithm

To form a solution, an ant starts at a random codeword then move from codeword x to codeword *y* based on the following probability function:

$$p_{xy}^{k}(t) = \frac{(\tau_{xy}(t))^{\alpha} \cdot (\delta_{xy})^{\beta}}{\sum_{l \in N_{kx}} (\tau_{xl}(t))^{\alpha} \cdot (\delta_{xl})^{\beta}} \tag{5}$$

where, α and β are influence parameters, δ_{xy} represents the attractiveness, which is the distance between the codewords x and y, τ_{xy} is the pheromone on the edge between the two codewords x and y, N_{kx} is the set of codewords feasible to take with codewords x by ant k. In our implementation all ants use the same definition of feasibility, which is being at distance $\delta \geq h$, where h is 3, 4, or 5. For every edge pairing x and y codewords, the pheromone update formula is:

$$\tau_{xy} = (1-\rho)\ \tau_{xy} + \sum_{m} F / F_{max} \tag{6}$$

where, ρ is the pheromone evaporation rate, *F* is the fitness of the solution and F_{max} is the maximum fitness so far. The second part updates only the edges of nodes that are part of the solution. The daemon actions here refer to some centralized housekeeping actions.

Experiments

We used the CRO ToolBox by Li and Lam (2015) as a ready framework for minimization and wrote Java code for the maximally distant codes allocation problem, while the whole ACO was written in C++ to allow using effective string manipulation of large structures. To test the power of the algorithms in directing the search with balanced objective, we ran several tests to locate sets of 8, 16 and 32 codewords using 8-bit strings. Varying μ from 0.0 to 10 in steps of 0.25, The CRO outperformed the ACO in both mean and minimum distance measure.

Table 3. CRO-distance against μ (8-bit Codewords)

	μ = 0.0		μ = 0.5		μ = 1.0	
8-bit	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

Table 4. ACO-Distance against μ (8-bit Codewords)

	μ = 0.0		μ = 0.5		μ = 1.0	
Set size	Mean	Min	Mean	Min	Mean	Min
8	4.29	2	4.54	3	4.57	1
16	3.42	1	4.24	2	4.27	1
32	2.57	1	4.10	1	4.13	1

Table 5. Performance against set size (8-bit Codewords)

	CRO		ACO	
Set size	Mean	Min	Mean	Min
8	4.32	4	4.21	3
16	4.25	3	3.97	2
32	4.08	2	4.08	1

Table 6. Performance against codeword length/set size

	CRO		ACO	
Length/set size	Mean	Min	Mean	Min
8-bit/8	4.32	4	4.21	3
10-bit/32	5.11	3	4.98	2
12-bit/128	6.04	2	5.75	1

Table 3 and 4 list those results, after discarding μ=0.25 and μ=0.75 as the results were quite similar to those of μ=0.5 in the two tests.

Table 5 shows the performance of both algorithms in locating sets of 8-bit Codewords with three sizes. The CRO outperformed the ACO marginally in the mean distance, but have always shown superiority in the minimal distance measure.

Table 6 shows the performance of both algorithms in finding three sets of fixed size and codeword length. The CRO outperformed the ACO marginally in the mean minimal distance. The reason is possibly that the ACO needs some parameter tuning, like the pheromone evaporation factor and the neighborhood threshold relation with the codeword length and set size.

Figure 4 and 5 show the progress of the CRO algorithm; the quality of the allocated sets over time represented by the number of evaluations or actions.

Tracing the progress of the CRO algorithms indicated that the first few thousands evaluations get more than 95% of the work done. To test the validity of this statement, we picked 100,000 solutions at random and started with the worst 100 solutions as initial population. The outcome was almost the same. The decomposition (A1T2) and combination (A2T1) condition were set to limit the rates to nearly 5 to 10% of the total evaluations for each.

Tests were run on i7 Intel dual core processor based desktops with 16 GB DRAM. Typical runs took few minutes for small instances. The two algorithms were given long enough time to run and the competition was in the ability of each to find a better solution rather than how fast the solution is found.

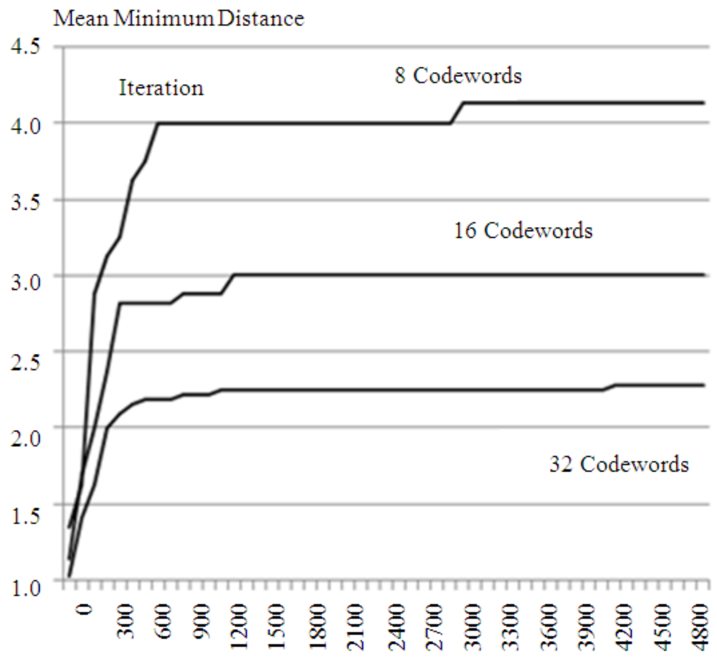

Fig. 4. CRO progress in locating 8-bit Codeword sets

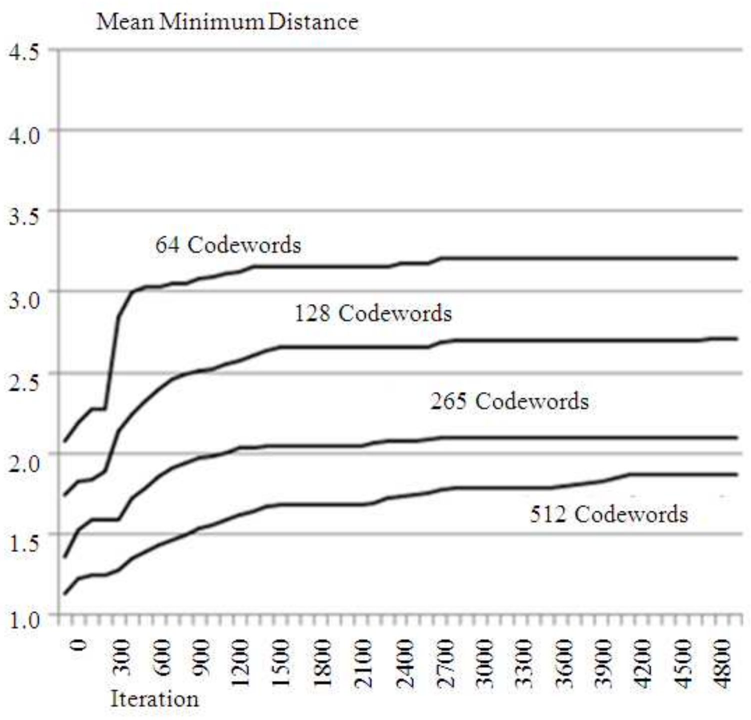

Fig. 5. CRO Progress in locating 12-bit codeword sets

Conclusion

Chemical Reaction Optimization and Ant Colony Optimization algorithms worked well in finding good solutions to the Maximally Distant Codes Allocation problem. The CRO outperformed the ACO marginally in the mean distance and drastically in the minimum distance in every test, regardless of the problem instance and time budget, even with various random initial populations. The reason for demonstrating lower performance is possibly due to the fact that the ACO algorithm parameters are a bit harder to tune than those of the CRO. To benchmark the CRO algorithm better, we plan to change the approach to find the largest set of codewords with preset minimum distance.

Acknowledgement

Thanks to Prince Sattam bin Abdulaziz University, Alkharj-Saudi Arabia and Jordan University of Science and Technology, Irbid-Jordan, for supporting this research.

Funding Information

This work was funded by the Deanship of Scientific Research at Prince Sattam bin Abdulaziz University, Alkharj-Saudi Arabia, which offered financial support under the research project No. 2014/01/1649.

Author’s Contributions

Taisir Eldos: Lead the team and assigned tasks to the team members, reviewed literature, proposed the mapping, analyzed the results and wrote the draft.

Waleed Nazih: Wrote code for mapping and actions and cost evaluation and ran the tests and produced results for the CRO algorithm, reviewed and revised the draft.

Aws Kanan: Wrote code for mapping and actions and cost evaluation and ran the tests and produced results for the ACO algorithm, reviewed and revised the draft.

Ethics

This research builds on previous work by the authors in which the same team investigated the use of the chemical reaction optimization in solving other complex engineering problems; the printed circuit board drilling.

References

Alba, E. and F. Chicano, 2004. Solving the error correcting code problem with parallel hybrid heuristics. Proceedings of the ACM Symposium on Applied Computing, Mar. 14-17, ACM, NY., USA, pp: 985-989.
DOI: 10.1145/967900.968101

Baterina, A.V. and C. Oppus, 2010. Image edge detection using ant colony optimization. WSEAS Trans. Signal Processing, 2: 58-67.

Bland, J.A., 2007. Local search optimisation applied to the minimum distance problem. J. Adv. Eng. Informatics Archive, 21: 391-397. DOI: 10.1016/j.aei.2007.01.002

Cotta, C., 2004. Scatter search and memetic approaches to the error correcting code problem. Proceedings of the Evolutionary Computation in Combinatorial Optimization, 4th European Conference, Apr. 5-7, Springer Berlin Heidelberg Coimbra, Portugal, pp: 51-61. DOI: 10.1007/978-3-540-24652-7_6

Dorigo, M. and L.M. Gambardella, 1997. Ant colonies for the travelling salesman problem. Biosystems, 43: 73-81. DOI: 1016/S0303-2647(97)01708-5

Eldos, T., A. Kanan and A. Aljumah, 2013a. Adapting the ant colony optimization algorithm to the printed circuit board drilling problem. World Comput. Sci. Inform. Technol. J., 3: 100-104.

Eldos, T., W. Nazih and H. Elsimary, 2013b. Error-correction-code allocation using the chemical reaction optimization algorithm. Int. J. Eng. Comput. Sci., 13: 54-57.

G´omez, O., 2005. Omicron ACO. A new ant colony optimization algorithm. CLEI Electr. J., 8: 1-8.

Ghosh, S., S. Basu and N.A. Touba, 2005. Selecting error correcting codes to minimize power in memory checker circuits. J. Low Power Electronics, 1: 63-72. DOI: 10.1166/jolpe.2005.007

Ginidi, A.R.G., M.A.M. Kamel and H.T. Dorrah, 2010. Development of new fuzzy logic-based ant colony optimization algorithm for combinatorial problems. Proceedings of the 14th International Middle East Power Systems Conference, Dec. 19-21, Cairo University, Egypt, pp: 331-338.

Haas, W. and S. Houghten, 2007. A comparison of evolutionary algorithms for finding optimal error-correcting codes. Proceedings of the 3rd IASTED International Conference on Computational Intelligence, (CCI'07), ACM, Anaheim, CA., pp: 64-70.

Ho, S.L., S. Yang, G. Ni, J.M. Machado, 2006. A modified ant colony optimization algorithm modeled on tabu-search methods. IEEE Trans. Magnet., 42: 1195-1198. DOI: 10.1109/TMAG.2006.871425

Hung, K.S., S.F. Su and Z.J. Lee, 2007. Improving ant colony optimization algorithms for solving traveling salesman problems. J. Adv. Comput. Intelligence Intelligent Informat., 11: 433-442.

Hwanga, W., C. Ou, C. Hsu and T. Lo, 2005. Iterative optimization for joint design of source and channel codes using genetic algorithms. J. Chinese Institute Eng., 28: 803-810. DOI: 10.1080/02533839.2005.9671050

Kanan, A., T. Eldos and M. Alkahtani, 2013. Mobile Ad Hoc networks routing using ant colony optimization. World Comput. Sci. Inform. Technol. J., 3: 105-109.

Lacan, J. and P. Chatonnay, 1999. Search of optimal error correcting codes with genetic algorithms. Proceedings of the 6th Fuzzy Days Dortmund, Theory and Applications International Conference, May 25-28, Springer, Germany, pp: 93-98. DOI: 10.1007/3-540-48774-3_12

Lam, A.Y.S. and V.O.K. Li, 2010a. Chemical reaction optimization for cognitive radio spectrum allocation. Proceedings of the IEEE Global Telecommunications Conference, Dec. 6-10, IEEE Xplore Press, Miami, FL., pp: 1-5. DOI: 10.1109/GLOCOM.2010.5684065

Lam, A.Y.S., J. Xu and V.O.K. Li, 2010. Chemical reaction optimization for population transition in peer-to-peer live streaming. Proceedings of the IEEE Congress on Evolutionary Computation, Jul. 18-23, IEEE Xplore Press, Barcelona, pp: 1-8. DOI: 10.1109/CEC.2010.5585933

Lee, H. and E. Kim, 2008. A symbiotic evolutionary design of error-correcting code with minimal power consumption. ETRI J., 30: 799-806. DOI: 10.4218/etrij.08.0108.0188

Li, V. and A. Lam, 2015. Chemical reaction optimization.

Lorpunmanee, S., M.N. Sap, A.H. Abdullah and C. Chompoo-inwai, 2007. An ant colony optimization for dynamic job scheduling in grid environment. Int. J. Comput., Control, Quantum Inform. Eng., 1: 1328-1335.

McCarney, D.E., S. Houghten and B.J. Ross, 2012. Evolutionary approaches to the generation of optimal error correcting codes. Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation, Jul. 07-11, ACM, Philadelphia, pp: 1135-1142. DOI: 10.1145/2330163.2330320

Rais, H.M., Z.A. Othman and A.R. Hamdan, 2007. Improved Dynamic Ant Colony System (DACS) on symmetric traveling salesman problem. Proceedings of the International Conference on Intelligent and Advanced Systems, Nov. 25-28, IEEE Xplore Press, Kuala Lumpur, pp: 43-48. DOI: 10.1109/ICIAS.2007.4658345

Shieh, C.S., J.S. Pan, C.J. Su and B.Y. Laio, 2003. Noise suppression for shape-gain vector quantization by index assignment using ant colony systems. Proceedings of the Joint Conference of the Fourth International Conference on Information, Communications and Signal Processing, 2003 and Fourth Pacific Rim Conference on Multimedia, Dec. 15-18, IEEE Xplore Press, pp: 235-238. DOI: 10.1109/ICICS.2003.1292450

Wolpert, D.H. and W.G. Macready, 2002. No free lunch theorems for optimization. IEEE Trans. Evolutionary Algorithms, 1: 67-82. DOI: 10.1109/4235.585893

Xu, J., A.Y.S. Lam and V.O.K. Li, 2010. Chemical reaction optimization for the grid scheduling problem. Proceedings of the IEEE International Conference on Communications, May 23-27, IEEE Xplore Press, Cape Town, pp: 1-5. DOI: 10.1109/ICC.2010.5502406

Yu, B., Z. Yang, C. Cheng and C. Liu, 2005. Optimizing bus transient network with parallel ant colony algorithm. Proc. Eastern Asia Society Transportation Stud., 5: 374-389.

Zaferanieh, M., T. Moallem and T. Sabzevar, 2009. Ant colony and simulated annealing algorithms for finding the core of a graph. World Applied Sci. J., 7: 1335-1341.