A Molecular Dynamics Heuristic for Solving the Traveling Salesperson Problem

Jaderick P. Pabico[1], Jose Rene L. Micor[2] and Ma. Christine A. Gendrano[3]
[1]*jppabico@uplb.edu.ph*, [2]*jrlmicor@uplb.edu.ph*, [3]*ma.christine.gendrano@dlsu.edu.ph*
[1]Institute of Computer Science, University of the Philippines Los Baños
[2]Institute of Chemistry, University of the Philippines Los Baños
[3]College of Computer Studies, De La Salle University – Science and Technology Complex

Abstract – *In this paper, a nature-based metaphor for computation is presented as a heuristic solution for a popular combinatorial optimization problem, the traveling salesperson problem (TSP). The metaphor was aptly named artificial chemistry (ACHEM) because the computational process is based on molecular dynamics. It is designed as a distributed stochastic algorithm that simulates reaction systems of algorithmic objects whose behavior is inspired by natural chemical systems. Finding the optimal solutions for TSP are particularly intractable for problem instances that are very large. This is the reason why a heuristic, such as the ACHEM, is a preferred solution than a computational procedure that provides optimal ones. To evaluate the utility of the heuristic, ACHEM was applied to find near-optimal solutions to large instances of the TSP. Results show that ACHEM outperformed other nature-based heuristics such as the simulated annealing and the self organizing maps, while it performed as good as the genetic algorithm and the ant colony optimization. Thus, ACHEM provides another natural metaphor for solving hard instances of the TSP.*

***Keywords* –** Artificial chemistry, combinatorial optimization, traveling salesperson problem, TSP

I. Introduction

The traveling salesperson problem (TSP) has been used as a paradigm for solving real-world problems such as shop floor control, scheduling, distribution of goods and services, vehicle routing, product design, and VLSI layout [1]. Given a set of cities, and known distances between each pair of cities, the TSP is the problem of finding a Hamiltonian tour such that the total distance traveled is minimum. A Hamiltonian tour is a tour that visits each city exactly once. TSP may also be stated as the search for the minimum Hamiltonian cycle instead, which is actually a Hamiltonian tour with the requirement that the salesperson return to the city where it started. Other TSP variants consider the cost of traveling between two cities, or the time it will take to travel between them, but the problem does not change [2],[3].

Exact solutions to solving TSP have been proposed by many researchers but these solutions are only efficient for small problem instances. TSP has proved to be intractable for large problem instances, where intractability of a solution means that even the fastest known computer will take a very long time to solve the problem. The TSP is intractable because if there are n cities, the number of possible tours is $(n - 1)!/2$. If, for example, the recent fastest computer can compute for the cost of one tour in 12μs, checking all possible tours when n is very large might take more than a human's lifetime. Table 1 shows the number of all possible tours and the approximate amount of time it will take to solve the TSP for some $n \leq 20$. Realistically, the amount of time to compute for the cost of one tour increases as the length of tour increases, which in turn increases as the number of cities (n) increases. To simplify the estimate, the approximate time in Table 1 did not take into account the corresponding increase in computing for the cost of one tour at $n>5$. It is highly possible that the values in Table 1 will take longer than estimated at problem instances where $n>5$. Most real-world applications that use TSP as a paradigm for computation have $n >> 20$. Thus, checking all possible solutions when $n > 20$ is impractical.

Table 1. The number of all possible tours and the approximate amount of time to solve n-city TSP's, where $n \leq 20$, with the assumption that a computer can compute for one tour in a constant time of 12μs.

Number of Cities (n)	Number of Possible Tours	Time
5	12	12 μs
8	2,520	2.5 ms
10	181,440	0.18 s
12	19,958,400	20 s
15	87,178,291,200	12.1 hours
18	177,843,714,048,000	5.64 years
20	60,822,550,204,416,000	1,927 years

Intractable problems are said to belong to the class of NP-hard problems and TSP has proved to belong to the same class [4],[5]. Because of the nature of the TSP, researchers have developed heuristic and metaheuristic methodologies so that intractable instances of the TSP may be given practical solutions. Practicality here means that the problem can be computed within a reasonable amount of time, while the solutions found are near-optimal. Computing within reasonable amount of time means that a satisfiable solution can be obtained within a specified deadline (which, intuitively, should be shorter than a human's lifetime), while near-optimality means that the seeker of the solution is already satisfied with the best solution found so far (which might not necessarily mean the very best solution, but the solution seeker has already found it practically useful anyway).

Heuristic solutions to TSP have been studied extensively. Graph-based heuristics such as branch and bound [6], cutting planes [7], Lagrangian relaxation [8],[9], and branch-and-cut [10], as well as multi-agent-based and nature-inspired algorithms such as genetic algorithms [11], memetic algorithms [12]–[14], tabu search [15], simulated annealing [16], simulated jumping [17], neural networks [18], and ant colonies [19]–[21] have been used and shown to find optimal and near optimal solutions to several instances of TSP.

In recent years, the chemical metaphor, called artificial chemistry (ACHEM), has emerged as a computational paradigm for search, optimization, and machine learning [22]–[27], which are useful computational tools in artificial intelligence and computer science. The processes in molecular dynamics have been used as a computational paradigm because chemical and biochemical systems of living organisms have been shown recently to possess computational properties [28]–[30]. This prompted researchers to develop a metaheuristic algorithm based on chemical dynamics. In this “kind” of chemistry, the objects (atoms or molecules) are paradigms for data or solutions to problems, while the interactions (collisions or reactions) among objects are paradigms for computation. The objects and their interactions to one another were used in the past to solve several *toy* problems such as the generation of prime numbers, robot control [27], and number division [31].

In this current effort, ACHEM was employed to search for near-optimal solutions to large instances of TSP. This was done by the following procedure:

1. Mapping Hamiltonian tours as artificial molecules;
2. Defining the cost of traversing the tours as molecular mass; and
3. Developing reactions as functions for creating solutions to TSPs from a randomly generated molecules in an occasionally-stirred reaction tank.

With these metaphors, optimal and near-optimal solutions to TSP were obtained through a method that has the same efficiency as the known multi-agent-based heuristics.

This paper introduces ACHEM and its utility as a computational metaphor for solving the TSP. Here, ACHEM is presented as a distributed approach to combinatorial optimization based on the dynamics of natural chemical systems. This presentation discusses ways of how information can be created and be processed by a collection of artificial molecules floating in a simulated reactor tank. ACHEM is shown here to search for the near-optimal solution to TSP through mapping of molecules to Hamiltonian tours, relating molecular mass to molecule’s rate of reaction, and developing the reaction algorithm.

II. Algorithm Development

This section briefly reviews the TSP and introduces ACHEM. The discussion proceeds to the development of algorithms that mimic chemical reactions that result in information processing. The processing of information happens in artificial reactor guided by reaction rules. As this simulation is inspired by the concepts of chemistry, popular nomenclature in the chemical sciences were extensively used. Readers are cautioned, however, that they are analogical only.

A. Traveling Salesperson Problem

TSP is defined as the problem of finding the shortest tour or cycle of a graph $G(V, E)$ that visit each vertex $v_i \in V$ once, $\forall i=1, ..., n$, and $n = |V|$. Formally, given a set of cities $V = \{v_1, v_2, ..., v_n\}$, an edge set $E = \{(i, j): v_i, v_j \in V\}$ representing roads that connect two cities, and a cost measure matrix C, where each matrix element $c_{i,j}$ is the cost measure associated with edge $(i, j) \in E$, TSP is the problem of finding the minimal Hamiltonian tour or cycle. Equations 1 and 2 show the costs of a Hamiltonian tour ($\mathbf{cost}(\mathrm{H}_t)$) and a Hamiltonian cycle ($\mathbf{cost}(\mathrm{H}_c)$), respectively. Figure 1 shows a 4-city TSP showing two cycles, χ_1 and χ_2, as possible solutions. In this example, χ_1 entails passing through vertices A, B, C, D, and back to A in that order, while χ_2 passes through vertices A, C, D, B, and A. With this example, it is easy to see that χ_1 costs 97, which is cheaper to χ_2 which costs 108. It is easy to see also that cycles A-B-C-D-A and B-C-D-A-B have the same costs but differ in their respective starting cities.

$$\mathbf{cost}(\mathrm{H}_t) = \Sigma_{i=1..n-1}\, c_{i,i+1} \quad (1)$$

$$\mathbf{cost}(\mathrm{H}_c) = c_{n,1} + \mathbf{cost}(\mathrm{H}_t) \quad (2)$$

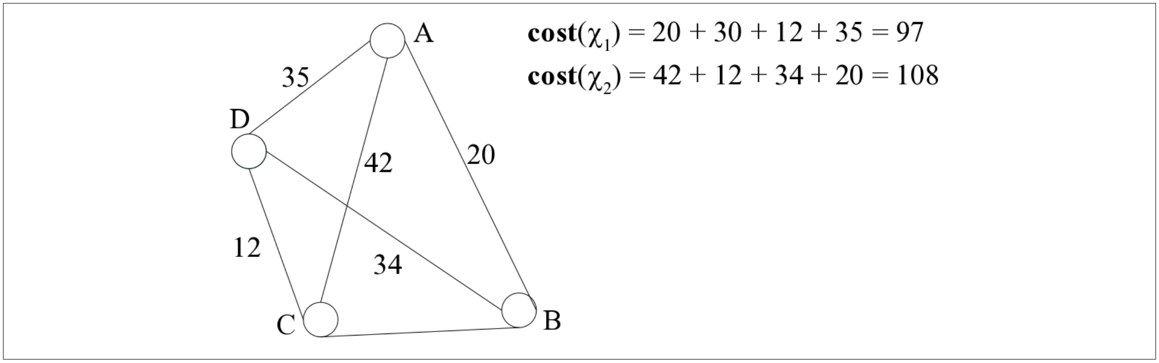
A
B
C
D
35
42
20
12
34
$\mathbf{cost}(\chi_1) = 20 + 30 + 12 + 35 = 97$
$\mathbf{cost}(\chi_2) = 42 + 12 + 34 + 20 = 108$

Fig. 1. An example 4-city TSP with two cycles χ_1 = A-B-C-D-A and χ_2 = A-C-D-B-A as possible solutions.

In cases where cities $v \in V$ are given by their coordinates (x, y) and $c_{i,j}$ is the Euclidean distance between cities v_i and v_j, then the problem is a Euclidean TSP. If $c_{i,j} = c_{j,i}$, then the problem is a symmetric TSP. If $c_{i,j} \neq c_{j,i}$ for at least one $c \in C$, then the problem is an asymmetric TSP. Other TSP instances are TSP(1,2), fractal [32], *k*-template, prize-collecting, circulant [33], on-line [34], time-dependent, angular-metric [35], maximum/minimum latency [36]–[38], bipartite [37], remote [39], and precedence-constrained that have also attracted considerable research attention in recent years. The test bed problems used in this contribution are the symmetric and asymmetric TSP instances from TSPLIB [40].

B. Artificial Chemistry

In the physical world, molecular interactions and their corresponding chemical reactions happen under specific physical and structural conditions. Molecules carry with them information specific to their composition such as molecular weight and molecular structure. Chemical reactions, on the other hand, cause changes to the composition of the reacting molecules. A change in composition means a change in the information being carried by the molecules. With this idea in mind, the composition of molecules can be seen as a kind of information storage, while the reaction between them as a kind of information processing. The more molecules involved in the reaction and the faster the reaction, the more information is processed. Therefore, one can create an abstract system, similar to chemical systems, which is capable of information storage and processing.

Formally, ACHEM is defined by a triple (M, R, A), where M is a set of artificial molecules, R is a set of reaction rules describing the interaction among molecules, and A is an algorithm driving the ACHEM system. The molecules in M may be composed of abstract symbols [41], strings of characters [42]–[44], λ–expressions [23], binary strings [27],[45], numbers [26], or logical or mathematical proofs [46]. This work introduces Hamiltonian cycles as molecules that encode solutions to TSP.

The rules in R can be defined explicitly [41] or implicitly by using string matching and string concatenation [42],[44],[47], λ–calculus [23],[48], Turing machines [28], finite state machines or machine language [27], proof theory [48], matrix multiplication [45], or simple arithmetic operations [26]. In this paper, reaction rule is presented as a reordering procedure. In this rule, when two molecules collide or interact, a new pair of molecules, different from the originally colliding molecules, is created, but both encoding portions of the reactants. It is possible that the products of the collision are the same as the reactants. When this happens, the collision is said to be elastic.

The algorithm A describes how the rules are applied to a “soup” of artificial molecules. The algorithm may simulate a well-stirred abstract topology-less reaction tank [23],[27],[42], an Euclidean discrete reaction vessel [41],[47], a continuous 3-dimensional space [49], or a self-organizing topology [50]. In this effort, A simulates a topology-less reaction tank that partitions the soup into levels of reaction activities as a function of molecular mass.

C. ACHEM for TSP

1) *Molecular Properties*

The vertices $v_i \in V$, $\forall i$=1, ..., n are considered as the set of atoms in the n-city TSP abstract world. These atoms exist in stable molecular forms that can be considered as Hamiltonian cycles. The set of artificial molecules M is the set of Hamiltonian cycles that visit the nodes $v_i \in V$ once, $\forall i$=1, ..., n. Each of the molecules $m \in M$ is a fixed-length n–ary string, where m is defined by a n-long regular expression $\{1|2| \ldots |n\}^n$ with the constraint that m contains only the n permutation of cities taken n. This constraint provides assurance that m encodes a valid Hamiltonian cycle. The cost of traversing the Hamiltonian cycle (Equation 2) is a function of the cost matrix C and can be regarded as the molecular mass of m. The molecular mass is directly proportional to the excitation energy of the molecule.

2) *Artificial Reactions*

Two reaction rules were designed: a zero-order reversible (i.e., non-catalytic) reaction (Equation 3) and a forward catalytic reaction (Equation 4). The first reaction can be considered as a collision of two molecules. All collisions of two molecules m_1 and m_2 may have unique outcomes, m_3 and m_4. Each collision can be represented as a function R_1: $M \times M \rightarrow M \times M$. However, if the products of the reaction are the same as the reactants, i.e., $m_1 + m_2 \Leftrightarrow m_1 + m_2$, then the collision is an elastic collision [51],[52].

$$m_1 + m_2 \Leftrightarrow m_3 + m_4 \qquad (3)$$

$$m_5 + w \Rightarrow m_6 \qquad (4)$$

Similar to the cycle crossover in genetic algorithms [53], the first reaction rule performs reordering under the constraint that each city comes from one reactant or the other. The reaction rule in Equation 3 is described in Algorithm 1:

Algorithm 1. Non-Catalytic Reaction Rule

```
1. Let an integer $l \in [1, n]$ be the index of the city encoded in
   any molecule $m$. The indexing order does not matter (i.e.,
   whether the index goes from left to right or vice- versa) as
   long as there is consistency in using one order of direction
   throughout this algorithm.
2. Take a random integer between 1 and $n$ and assign it to $l$.
   Let $l_0$ = l.
3. Taking the reactant $m_1$, locate the $l$th atom in $m_1$ and move it
   as the $l$th atom for $m_3$.
4. Take note of the $l$th atom in $m_2$ and locate it in $m_1$. Replace
   the value of $l$ with the index of the atom found in $m_1$.
5. Repeat steps 3 to 4 until the $l$th atom in $m_2$ is the same as
   the $l_0$th atom in $m_1$.
6. For all indeces l with no atoms yet in $m_3$, move the $l$th atom
   from reactant $m_1$ as the $l$th atom in product $m_3$.
7. Repeat steps 2 to 6 for reactant $m_2$ and product $m_4$.
```

An elastic collision of the form $m_1 + m_2 \Leftrightarrow m_1 + m_2$ happens when the stopping criterion described in step 5 of the above rule is reached during the first iteration. Consider for example a 10-city TSP with two molecules m_1 encoding the cycle I-H-B-A-G-D-E-J-F-C-I and m_2 with cycle a-b-c-d-e-f-g-h-i-j. Notice that the lower case letter was used in m2 for illustration purposes only so that one can see where the cities on thefinal product came from. Assume further that l=1 was used as obtained randomly in step 2 of Algorithm 1. Comparing the lth (i.e., first) city of both tours, it is seen that city I in m_1 is matched to city A in m_2, which is in the 4th position in m_1. The product m_3 will then take city I as its first city, and city A as its fourth city. City D is the 4th city in m_2, which is the 6th city in m_1. Thus, m_3 will take city D as its 6th city. Continuing this process, m_3 will soon contain the following cities in its cycle so far: I-*-*-A-*-D-*-*-F-*-J-I, where an * symbol means that the position has not been filled-up yet. Notice that the filled-up positions are cities that came from m_1. To fill up the unfilled positions in m_3, respective positions in m_2 are copied in m_3, resulting in the following cycle for m_3: I-b-c-A-e-D-g-h-F-j-I. Consequently, m_4 will encode the following: a-H-B-d-G-f-E-J-i-C-a. Notice that both products m_3 and m_4 are valid Hamiltonian cycles. Figure 2 summarizes this process in a simple colored visualization.

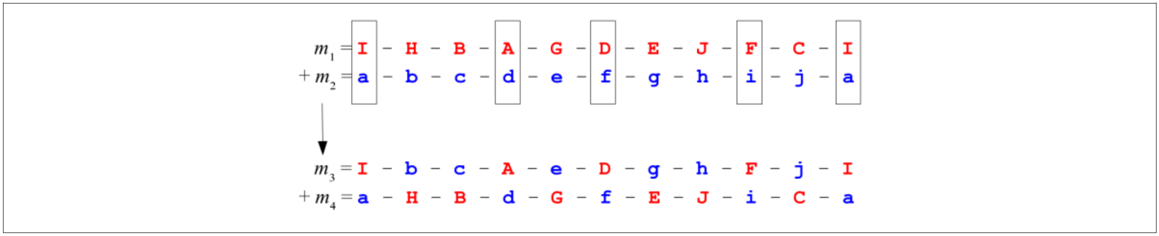
m_1 = I - H - B - A - G - D - E - J - F - C - I
$+ m_2$ = a - b - c - d - e - f - g - h - i - j - a
m_3 = I - b - c - A - e - D - g - h - F - j - I
$+ m_4$ = a - H - B - d - G - f - E - J - i - C - a

Fig. 2. Two cycles m_1 and m_2 colliding according to Algorithm 1 to produce m_3 and m_4.

The second reaction (Equation 4) can be considered as a collision of molecule m_5 with the tank's walls w to produce another molecule m_6. The wall w can be considered as a catalyst of the reaction and is never modified, thus this reaction can be represented by a function R_2: $M \rightarrow M$ and is algorithmically described in Algorithm 2 with simple colored visualization in Figure 3 for a 10-city TSP and random l=5.

Algorithm 2. Catalytic Reaction Rule

```
1. Let an integer $l \in [1, n]$ be the index of the city encoded in molecule $m_5$.
2. Take a random integer between 1 and $n - 1$ and assign it to $l$. $l$ will represent the $l$th atom of $m_5$
   that will collide with $w$.
3. Swap the $l$th atom with the $(l + 1)$th atom. The resulting molecule will be $m_6$.
```

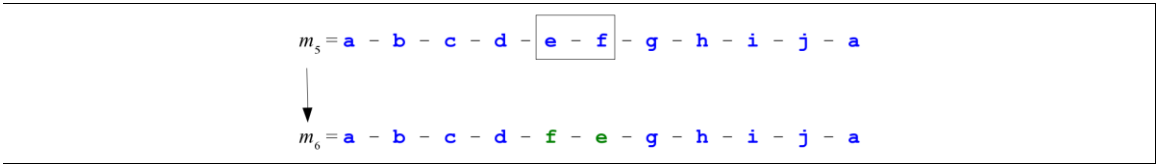
m_5 = a - b - c - d - e - f - g - h - i - j - a
m_6 = a - b - c - d - f - e - g - h - i - j - a

Fig. 3. A sample cycle m_5 colliding with the wall according to Algorithm 2 to produce m_6.

3) *Reaction Vessel*

The reactor algorithm A operates on a *soup* of molecules $S = \{m_1, ..., m_{||S||}\}$, $||S|| << ||M||$. The development of the soup S is realized by iteratively applying steps 2 to 5 of Algorithm 3.

Algorithm 3. Reactor Algorithm

```
1. Initialize the soup $S$ with $||S||$ molecules selected randomly from $M$.
2. Using stochastic sampling with replacement, select two molecules $m_1$ and $m_2$ from $S$ without removing them.
3. Apply the reaction rule in Algorithm 1 for the two reactants $m_1$ and $m_2$ to produce the products $m_3$ and $m_4$.
4. With a small chance, apply the reaction rule in Algorithm 2 for a randomly chosen heavy (i.e. higher mass) molecule
   $m_5$ to produce $m_6$.
5. Repeat steps 2 to 3 until the density of molecules with lower molecular mass exceeds a tolerance level.
6. Decay the heavier molecules by removing them out of $S$ and replace them with randomly selected molecules from $M$.
7. Go back to step 2 unless the density of molecules with lower molecular mass did not improve.
8. What remains is a soup of molecules that encode optimal or near-optimal solutions to TSP.
```

The sampling procedure of step 2 of Algorithm 3 gives higher chances of reacting or colliding to molecules with low molecular masses. This simulates the level of excitation energy a molecule needs to overcome so that it will be able to collide with another molecule easily. This means that the lighter the molecule, the higher the chances that it will collide with other similarly-lighter molecules. The parallelism between ACHEM and real chemistry and summary of notations used are summarized in Table 2. These algorithms were applied to find solutions to large instances of symmetric and asymmetric TSPs.

Table 2. Parallelism of real chemistry and ACHEM and summary of notations used.

Real Chemistry	ACHEM	Notation
Set of atoms	Set of vertices	M
Single atom	A vertex	v
A molecule	A Hamiltonian cycle	m
Molecular mass	Tour cost	f
Reaction	Algorithms 1 and 2	R_1 and R_2
Universe	Soup	S
Chemistry	Algorithm 3	A

III. Results and Discussion

A. ACHEM Performance

ACHEM was applied to solve five sets of random instances of symmetric 50–city TSPs, and to four examples of asymmetric TSPs namely, Oliver30 (a 30–city problem), EIl50 (a 50–city problem), EIl75 (a 75–city problem), and KroA100 (a 100–city problem) [40]. These test problems were chosen because these were the ones used by other researchers. Because of this, one can compare the results obtained here with those obtained by other nature–inspired heuristics such as simulated annealing (SA) [54], self organizing maps (SOM) [55], genetic algorithms (GA) [56],[57], and ant colony optimization (ACO) [19].

Table 3 shows the comparison of the average tour length found by ACHEM, SA and SOM on five sets of random instances of symmetric 50–city TSPs.

Table 3. Comparison of the average tour length found by ACHEM, SA, and SOM on five sets of random instances of symmetric 50-city TSPs. Values in boldface are the best average tour length per problem.

Problem	ACHEM	SA	SOM
1	**5.87**	5.88	6.06
2	6.15	**6.01**	6.25
3	**5.59**	5.65	5.83
4	**5.67**	5.81	5.87
5	**6.15**	6.33	6.70

Values in boldface are the best average tour length for each of the problem sets. ACHEM results were averaged over 5 trials. It can be seen that ACHEM performs fairly as SA while it outperforms SOM.

Table 4 shows the comparison of the best integer tour length found by ACHEM, GA and ACO on four examples of asymmetric instances of TSPs. Values in boldface are the best average tour length for each of the problem instances. Results show that ACHEM has the same performance as both GA and ACO at lower problem instances ($n \leq 50$), and outperforms both GA and ACO at a higher problem instance ($n > 50$).

Table 4. Comparison of the best integer tour length found by ACHEM, GA, and ACO on four examples of asymmetric instances of TSP. Values in boldface are the better tour length found per problem.

Problem	ACHEM	GA	ACO
Oliver30	**421**	**421**	**421**
Eil50	**424**	428	**424**
Eil75	550	**545**	555
KroA100	**21,280**	21,761	22,363

B. Soup Development

Figure 4a shows the development of the soup in a reactor tank of ACHEM that solves a randomly created symmetric 10–City TSP of known optimum. Here, the optimum cost is **cost**(H_c) = 500. Notice further that the optimum Hamiltonian cycle was found at the third simulation epoch, lost it at the fourth epoch, and then rediscovered during the fifth epoch where it was never lost until the end of simulation run. One epoch means one iteration of applying steps 2 through 5 of Algorithm 3. Notice that the soup evolved into having molecules that store Hamiltonian cycles with lower **cost**(H_c)'s. The downward spikes in the maximum line (shown in Figure 4a with gray circles) show that during the simulation, the soup was almost saturated with molecules of lower molecular mass, as the difference between the maximum and minimum **cost**(H_c)'s is just 20. This observation was collaborated by Figure 4b which shows the tank's saturation level of molecules with lower molecular mass. The peaks in Figure 4b coincide with the downward spikes in the maximum line of Figure 4a. This signifies that ACHEM was able to find other Hamiltonian cycles of almost optimal costs during the simulation epoch where the peaks and spikes occur.

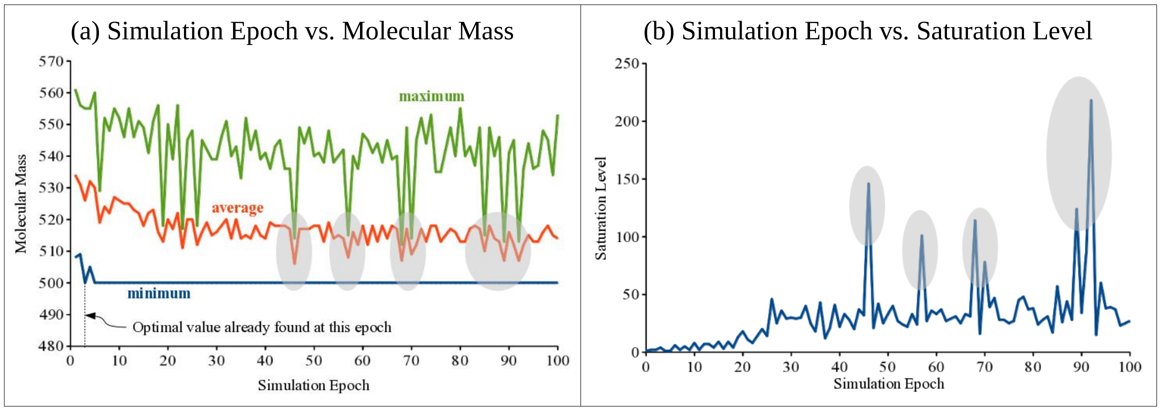

Fig. 4. The development of the soup in a reactor tank that solves a 10-city TSP.

Figure 5a shows the development of a soup of molecules designed to solve the KroA100, an example of a 100-city TSP. Here, the optimal tour has **cost**(H_c) = 5,200 and was found by ACHEM during its 92nd epoch. Notice that the peaks and spikes still occur but not as sharp as that of the 10-city TSP. Notice also that the evolution of the development of the non-optimal soup to the soup with optimal values follows a curvilinear pattern. This evolutionary progress is normal for higher problem instances. Figure 5b shows the saturation level of the best molecule at each epoch. Notice that at the 92nd epoch, the number of best molecules only counts to three, which is enought for ACHEM to find the optimal solution. Had ACHEM sustain the saturation count of 16 it obtained during the 5th epoch, it might have found the optimal earlier than the 92nd epoch. This lateness can be attributed to the effects of reversible reaction used in the simulation.

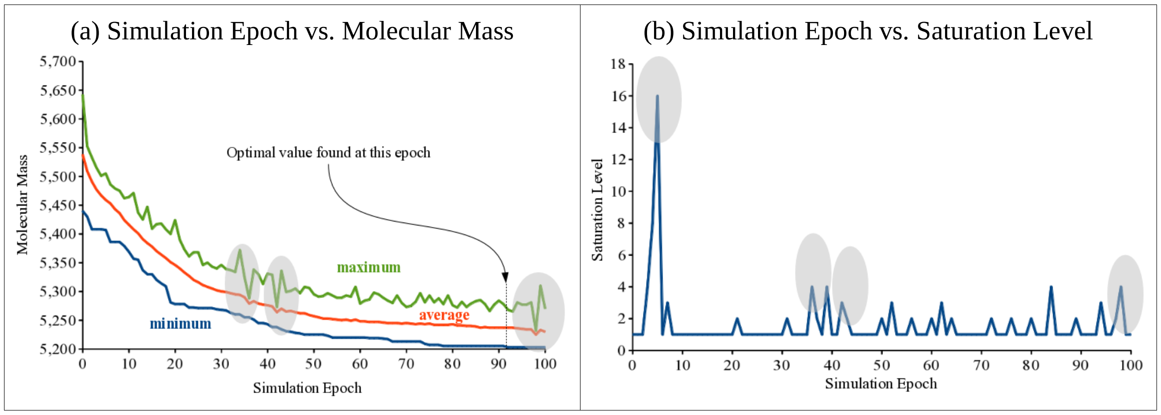

Fig. 5. The development of the soup in a reactor tank that solves a 100-city TSP.

C. Effects of Reversible Reaction

The sharpness of peaks in the saturation graphs presented in Figures 4b and 5b can be explained by the nature of the reaction rules used in the ACHEM simulations, particularly Algorithms 1 and 2. With these algorithms, a soup of molecules saturated with higher molecular mass at time T will alternately develop into a soup saturated with molecules of lower molecular mass at time $T+\Delta T$, and then into a soup of molecules with higher molecular mass at next time $T+2\Delta T$. The value of ΔT depends on the frequency of collisions among molecules, while the conservation of mass can be constantly assumed via some other way. This kind of behavior brought about by the algorithms in the soup of molecules may distrupt the evolution of molecules that encode the optimal Hamiltonian cycle, which as a result prolong the search.

To avoid the disrupting effects of the reaction rule on the solutions being solved by the artificial chemical system, it is recommended that the reaction rule be designed to keep the reactants to the product side of the reaction equation. For example, a second-order catalytic reaction rule of the form $x_1 + x_2 + X \Rightarrow x_1 + x_2 + x_3 + x_4$ that can initiate a mass-action kinetics might be a better one than the rule that was used in this study. Here, the concentration of the implicit substrate X might be kept constant. The production of new atoms x_3 and x_4 might create an implicit competition for space which may lead to an evolutionary process. This kind of reaction rule is already being studied as an extension of this research.

IV. Conclusion

In this paper, it was shown that artificial chemical objects can store information while the reactions among them can initiate information processing. An artificial chemical system that is capable of solving large instances of combinatorial optimization problems such as the TSP was designed. By giving computational metaphor to molecular properties as solutions and to molecular reactions as ways to create new solutions, the ACHEM system was able to develop an artificial soup of molecules in a reactor tank that store optimal and near-optimal solutions to TSP. Experiments showed that ACHEM can find quality solutions to TSP with the same quality as other nature-inspired methods.

Acknowledgements

This research effort is funded by and was conducted at the Research Collaboratory for Advanced Intelligent Systems, Institute of Computer Science, University of the Philippines Los Baños, College, Laguna.

References

[1] D.L. Applegate, R.E. Bixby, V. Chvatal and W.J. Cook. 2007. **The Traveling Salesman Problem: A Computational Study**. Princeton University Press: Princeton (ISBN 978069112993).

[2] M. Gendreau, J-Y. Potvin, O. Braysy, G. Hasle abd A. Lokketangen. 2008. *Metaheuristics for the vehicle routing problem and its extensions: A categorized bibliography*. In B.L. Golden, S. Raghavan and E.A. Wasil (eds.) **The Vehicle Routing Problem: Latest Advances and New Challenges. Operations Research/Computer Science Interfaces Series**, volume 42. Springer: New York, pp. 143–169 (ISBN 0387777776).

[3] K.L. Hoffman, M. Padberg and G. Rinaldi. 2013. *Traveling salesman problem*. In S.I. Gass and M.C. Fu (eds.) **Encyclopedia of Operations Research and Management Science**, 3rd edition. Springer:New York, pp. 1573–1578 (ISBN 978-1-4419-1137-7).

[4] R.M. Karp. 1972. *Reducibility among combinatorial problems*. In J.W. Thatcher and R.E. Miller (eds.) **Complexity of Computer Computations**. Plenum: New York. pp. 85–103 (ISBN 0306307073).

[5] M.R. Garey and D.S. Johnson. 1979. **Computers and Intractability: A Guide to the Theory of NP-Completeness**. Macmillan Higher Education:San Francisco, CA, pp. 338 (ISBN 978-0716710455).

[6] S. Tschoke, R. Luling and B. Monien. 1995. *Solving the traveling salesman problem with a distributed branch-and-bound algorithm on a 1024 processor network*. In **Proceedings of the 9th International Parallel Processing Symposium** (IPPS95), pp. 182–189.

[7] V. Chvatal, W. Cook, G.B. Dantzig, D.R. Fulkerson and S.M. Johnson. 2010. *Solution of a large-scale traveling-salesman problem*. In M. Junger, T.M. Liebling, D. Naddef, G.L. Nehhauser, W.R. Pulleyblank, G. Reinelt, G. Rinaldi and L.A. Wolsey (eds.) **50 Years of Integer Programming 1958 – 2008 : From the Early Years to the State-of-the-Art**. Springer-Verlag:Berlin. pp. 7–28 (DOI 10.1007/978-3-540-68279-0_1).

[8] M. Held and R.M. Karp. 1970. *The traveling salesman problem and minimum spanning trees*. **Operations Research** 18(6):1138–1162.

[9] M. Held and R.M. Karp. 1971. *The traveling salesman problem and minimum spanning trees: Part II*. **Mathematical Programming** 1(1):6–25.

[10] M. Padberg and G. Rinaldi. 1987. *Optimization of a 532-city symmetric traveling salesman problem by branch-and-cut*. **Operations Research Letters** 6:1–7.

[11] J.P. Pabico and E.A. Albacea. 2008. *The Interactive Effects of Operators and Parameters to GA Performance Under Different Problem Sizes*. **Philippine Computing Journal** 3(2):26–37 (ISSN 1908-1995).

[12] P. Moscato and M.G. Norman. 1992. *A memetic approach for the traveling salesman problem: Implementation of a computational ecology for combinatorial optimization on message-passing systems*. In M. Valero, E. Onate, M. Jane, J.L. Larriba and B. Suarez (eds.) **Parallel Computing and Transputer Applications 1: Proceedings of the International Conference**, Barcelona, Spain. IOS Press:Amsterdam, pp. 187–194 (ISBN 978-9051990966).

[13] B. Freisleben and P. Merz. 1996. *A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems*. In **Proceedings of the 1996 IEEE International Conference on Evolutionary Computation** (ICEC96), Nagoya University, Japan. IEEE: Piscataway, NJ, pp. 616–621 (ISBN 0-7803-2902-3).

[14] B. Freisleben and P. Merz. 1996. *New genetic local search operators for the traveling salesman problem*. In H.M. Voigt, W. Ebeling, I. Rechenberg, and H.P. Schwefel (eds.) **Proceedings of the 4th Conference on Parallel Problem Solving from Nature** (PPSN IV), Berlin, Germany, Lecture Notes in Computer Science Volume 1141, pp. 890–899. Springer (ISBN 978-3-540-61723-5).

[15] M. Zachariasen and M. Dam. 1996. *Tabu search on the geometric traveling salesman problem*. In I.H. Osman and J.P. Kelly (eds.) **Meta-Heuristics: Theory and Applications**. Springer: US, pp. 571–587 (ISBN 978-1-4613-1361-8).

[16] O.C. Martin and S.W. Otto. 1996. *Combining simulated annealing with local search heuristics*. **Annals of Operations Research** 63(1):57–75 (DOI 10.1007/BF02601639).

[17] S. Amin. 1999. *Simulated jumping*. **Annals of Operations Research** 86:23–38 (DOI 10.1023/A:1018954718550).

[18] O. Miglino, D. Menczer and P. Bovet. 1994. *A neuro-ethological approach for the TSP: Changing methaphors in connectionist models*. **Journal of Biological systems** 2(3):357–366. DOI 10.1142/S0218339094000210.

[19] L.M. Gambardella and M. Dorigo. 1995. *Ant-Q: A reinforcement learning approach to the traveling salesman problem*. In **Proceedings of the Twelfth International Conference on Machine Learning**, Tahoe City, CA. Morgan Kaufmann Publishers, pp. 252–260 (ISBN 1558603778).

[20] L.M. Gambardella and M. Dorigo. 1996. *Solving symmetric and asymmetric TSPs by ant colonies*. In **Proceedings of the 1996 IEEE International Conference on Evolutionary Computation** (ICEC96), Nagoya University, Japan. IEEE: Piscataway, NJ, pp. 622–627 (ISBN 0-7803-2902-3).

[21] M. Dorigo and L.M. Gambardella. 1997. *Ant colonies for the traveling salesman problem*. **BioSystems** 43(2):73–81 (DOI 10.1016/S0303-2647(97)01708-5).

[22] G. Berry and G. Boudol. 1992. *The chemical abstract machine*. **Journal of Theoretical Computer Science** 96(1):217–248 (DOI 10.1016/0304-3975(92)90185-I).

[23] W. Fontana. 1992. *Algorithmic chemistry*. In C.G. Langton, C. Taylor, J.D. Farmer and S. Rasmussen (eds.) **Proceedings of the Workshop on Artificial Life** (ALIFE90) 88:159–209, Redwood City, CA. Addison–Wesley.

[24] W. Banzhaf. 1995. *Self-organizing algorithms derived from RNA interactions*. In W. Banzhaf and F.H. Eeckman (eds.) **Evolution and Biocomputation, Lecture Notes in Computer Science** 899:69–102. Springer:Berlin (DOI 10.1007/3-540-59046-3_6).

[25] T. Ikegami and T. Hashimoto. 1995. *Coevolution of machines and tapes*. In F. Moran, A. Moreno, J.J. Merelo and P. Chacon (eds.) **Advances in Artificial Life: Proceedings of Third European Conference on Artificial Life, Lecture Notes in Computer Science** 929:234–245, Berlin. Springer–Verlag (DOI 10.1007/3-540-59496-5_302).

[26] W. Banzhaf, P. Dittrich and H. Rauhe. 1996. *Emergent computation by catalytic reactions*. **Nanotechnology** 7(1):307–314 (DOI 10.1088/0957-4484/7/4/001).

[27] P. Dittrich, W. Banzhaf, H. Rauhe and J. Ziegler. 1998. *Macroscopic and microscopic computation in an artificial chemistry*. In P. Dittrich, H. Rauhe and W. Banzhaf (eds.) **Proceedings of the Second German Workshops on Artificial Life** (GWAL97), University of Durtmond, pp. 19–22 (ISSN 0941-4568).

[28] A. Hjemfelt, E.D. Weinberger and J. Ross. 1991. *Chemical implementation of neural networks and Turing machines*. **Proceedings of National Academy of Sciences of the United States of America** 88(24):10983–10987.

[29] L.M. Adleman. 1994. *Molecular computation of solutions to combinatorial problems*. **Science** 266(5187):1021–1024 (DOI 10.1126/science.7973651).

[30] A. Arkin and J. Ross. 1994. *Computational functions in biochemical reaction networks*. **Journal of Biophysics** 67(2):560–578 (DOI 10.1016/S0006-3495(94)80516-8).

[31] P. Dittrich. 1998. *Real evolution in artificial chemistries*. In C.L. Nehaniv and G.P. Wagner (eds.) **The Right Stuff: Appropriate Mathematics for Evolutionary and Developmental Biology**, Technical Report Number 315, University of Hardfordshire School of Information Science, pp. 27–31.

[32] M. G. Norman and P. Moscato. 1995. *The Euclidean traveling salesman problem and a space-filling curve*. **Chaos, Solutions and Fractals** 6:389–397.

[33] Q. F. Yang, R. E. Burkard, E. Cela and G. J. Woeginger. 1995. *Hamiltonian cycles in circulant digraphs with two stripes*. **Technical Report SFB-Report 20, Technische Universitat Graz**.

[34] G. Ausiello, E. Feuerstein, S. Leonardi, L. Stougie and M. Talamo. 1994. *Server request with on-line routing*. In **Proceedings of the Fourth Scandinavian Workshop on Algorithm Theory** (SWAT94), Aarhus, Dinamarca, Springer–Verlag.

[35] A. Aggarwal, D. Coppersmith, S. Khanna, R. Motwani and B. Schieber. 1997. *The angular-metric traveling salesman problem*. In **Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms**.

[36] A. Blum, P. Chalasani, D. Coppersmith, B. Pulleybank, P. Raghavan and M. Sudan. 1994. *The minimum latency problem*. In **Annual Symposium on Theory of Computation** (STOC).

[37] P. Chalasani, R. Motwani and A. Rao. 1996. *Approximation algorithms for robot grasp and delivery*. In **2nd International Workshop on Algorithmic Foundations of Robotics** (WAFR).

[38] M.X. Geomans and J. Kleinberg. 1996. *An improved approximation ratio for the minimum latency problem*. In **Proceedings of 7th ACM–SIAM Symposium on Discrete Algorithms**.

[39] M.M. Halldorsson, K. Iwano, N. Katoh and T. Tokuyama. 1995. *Finding subsets maximizing minimum structures*. In **Proceedings of 6th ACM–SIAM Symposium on Discrete Algorithms**.

[40] G. Reinelt. 1991. *TSPLIB–a traveling salesman library*. **ORSA Journal on Computing** 3:376–384.

[41] F.J. Varela, H.R. Maturana and R. Uribe. 1974. *Autopoiesis: The organization of living systems*. **BioSystems** 5(4):187–196.

[42] R.J. Bagley and J.D. Farmer. 1992. *Spontaneous emergence of a metabolism*. In C.G. Langton, C. Taylor,

J.D. Farmer and S. Rasmussen (eds.) **Proceedings of the Workshop on Artificial Life** (ALIFE90), Volume 5 of Santa Fe Institute Studies in the Sciences of Complexity, pages 93–140, Redwood City, CA, Addison–Wesley.

[43] S.A. Kauffman. 1993. **The Origins of Order**. Oxford University Press.

[44] J.S. McCaskill, H. Chorongiewski, D. Mekelburg and U. Tangen. 1994. *Configurable computer hardware to simulate long-time self-organization of biopolymers*. **International Journal of Physical Chemistry** 98:1114–1115.

[45] W. Banzhaf. 1993. *Self-replicating sequences of binary numbers – Foundations I and II: General and strings of length n = 4*. **Biological Cybernetics** 69: 269–281.

[46] W. Fontana and L.W. Buss. 1996. *The barrier of objects: From dynamical system to bounded organizations*. In J.L. Casti and A. Karlqvist (eds.) **Boundaries and Barriers: On the Limits to Scientific Knowledge**, pp. 56– 116, Addison–Wesley, ISBN 0788196758.

[47] M. W. Lugowski. 1989. *Computational metabolism*. In C.G. Langton (ed.) **Artificial Life**.

[48] W. Fontana and L.W. Buss. 1994. *What would be conserved if the tape is played twice?* **Proceedings of the National Academy of Sciences of the United States of America** 91(2):757–761.

[49] K.P. Zauner and M. Conrad. 1996. *Simulating the interplay of structure, kinetics, and dynamics in complex biochemical networks*. In R. Hofestadt, M. Loffler, T. Lengauer and D. Schomburg (eds.) **Computer Science and Biology – Proceedings of the German Conference on Bioinformatics** (GCB96), IMISE Report, Universitat Leipzig, Germany, pp. 336–338.

[50] P. Dittrich and W. Banzhaf. 1997. *A topological structure based on hashing – Emergence of a spatial organization*. In P. Husbands and I. Harvey (eds.) **Fourth European Conference on Artificial Life** (ECAL97), University of Sussex, Brighton, UK, MIT Press, ISBN 0-262-58157-4.

[51] J.P. Pabico, J.R.L. Micor and E.R.E. Mojica. 2003. *Solving the scheduling problem using artificial chemistry*. In **32nd Annual Convention of the Kapisanang Kimika ng Pilipinas, Southern Tagalog Chapter**, ERDB Auditorium, University of the Philippines Los Baños.

[52] J.P. Pabico, E.R.E. Mojica and J.R.L. Micor. 2003. *Artificial chemistry: Basic concepts and application to combinatorial problems*. In **32nd Annual Convention of the Kapisanang Kimika ng Pilipinas, Southern Tagalog Chapter**, ERDB Auditorium, University of the Philippine Los Baños.

[53] D.E. Goldberg. 1989. **Genetic Algorithms in Search, Optimization and Machine Learning**. Addison–Wesley, Boston, MA.

[54] R. Durbin and D. Willshaw. 1987. *An analogue approach to the traveling salesman problem using an elastic net method*. **Nature** 326:689–691.

[55] J. Y. Potvin. 1993. *The traveling salesman problem: A neural network perspective*. **ORSA Journal of Computing** 5(4):328–347.

[56] D. Whitley, T. Starkweather and D. Fuquay. 1989. *Scheduling problems and traveling salesman: The genetic edge recombination operator*. In J.D. Schaffer (ed.) **Proceedings of the Third International Conference on Genetic Algorithms**, San Mateo, CA, Morgan Kaufmann, pp. 133–140.

[57] H. Bersini, C. Oury and M. Dorigo. 1995. *Hybridization of genetic algorithms*. **Technical Report IRIDIA 95–22, IRIDIA, Universite Libre de Bruxelles**, Belgium.