

Parallel chemical reaction optimization for utilization in intelligent RNA prediction systems

Helena Stegherr, Anthony Stein, Jörg Hähner

Angaben zur Veröffentlichung / Publication details:

Stegherr, Helena, Anthony Stein, and Jörg Hähner. 2019. "Parallel chemical reaction optimization for utilization in intelligent RNA prediction systems." In Intelligent Systems Workshop, part of ARCS 2019: 32nd International Conference on Architecture of Computing Systems; workshop proceedings; May 20- 21, 2019, Technical University of Denmark, Copenhagen, Denmark, edited by Carsten Trinitis and Thilo Pionteck, CD-ROM, 135-42. Berlin: VDE Verlag.
<https://ieeexplore.ieee.org/document/8836194>.



Parallel Chemical Reaction Optimisation for the Utilisation in Intelligent RNA Prediction Systems

Helena Stegherr, Anthony Stein, and Jörg Hähner

Organic Computing Group

University of Augsburg, Germany

helena.stegherr@student.uni-augsburg.de

{anthony.stein, joerg.haehner}@informatik.uni-augsburg.de

Abstract—Traditional, strictly deterministic algorithms are reaching their limits as modern information systems are often challenged with tremendously complex optimisation tasks. This can be in terms of reasonable computational effort or due to the lack of gradient information. Therefore, a necessity of designing Intelligent Systems becomes apparent and the increasing appearance of such systems substantiates the currently observable advent of Artificial Intelligence. Thus, Computational Intelligence methods, such as metaheuristics, gain entry into modern systems in order to render the challenging problems tractable again, sometimes with the downside of only converging to approximately optimal solutions. An approach inspired by chemical reactions which attracted rather less research attention so far, especially with regard to the utilisation in Organic Computing research, has been proposed recently. In this working paper, the Chemical Reaction Optimisation (CRO) algorithm is transformed towards a parallel variant, called pACRO, in order to decrease convergence time and scalability by exploiting multi-core computing architectures. The proposed parallel algorithm is empirically validated on well-known benchmark functions and set in relation to the findings reported in the literature. First results indicate that pACRO can compete with its predecessors. Furthermore, initial steps towards an application in the highly complex task of RNA Secondary Structure Prediction (RNA-SSP) are taken by outlining novel ways to construct the search space for helices as well as a solution candidate encoding based on which pACRO can be utilised.

Index Terms—Organic Computing, Population-based Metaheuristics, Chemical Reaction Optimisation, RNA Secondary Structure Prediction

I. INTRODUCTION

In light of the recent advances in *Artificial Intelligence* (AI) which allows machines to deal with highly complex tasks, naturally the imposed demands on *Information and Communication Technology* (ICT) systems also increase dramatically. The transfer of AI technology to systems deployed in the context of realistic and critical domains such as Healthcare, Traffic Control, Avionics and Autonomous Driving, to name a few, opens numerous potentials but also introduces challenges regarding the careful and safe design of these inevitably occurring *Intelligent Systems* (IS). Besides these specific design challenges regarding the increasing degree of autonomy of IS, the need for continual improvement, i.e. self-optimisation of such systems is a crucial requirement. Search and planning together constitute a major branch in AI research. Stochastic population-based search heuristics

have proven themselves very successful in various domains where the underlying search space is opaque and gradient-information are missing. Prominent representatives are *Evolutionary Algorithms* with their numerous derivatives, *Particle Swarm Optimisation* or *Ant-Colony Optimisation*. IS need to be self-adaptive in a sense to be able to quickly respond, i.e. re-optimize, in highly dynamic environments and their resulting ever-changing contextual observations on which predictions or even control decisions are based. One way to achieve such a progressing self-optimising behaviour is to incorporate such previously mentioned competent and efficient optimisation algorithms allowing for rapid responses to changing system states. But self-adaptation is not the only scenario where the efficiency of optimisation heuristics plays an important role. Real world applications from the domain of biology come with their idiosyncratic characteristics such as highly complex and vast search spaces due to e.g. epistatic gene interactions and heterogeneity. The identification of the most suitable candidates among the great variety of existing approaches and their purposeful improvement in view of their application in very specific tasks is a very important step on the way to reaching superhuman performance.

The need for IS involving AI technology in order to support biological engineers and geneticists to discover e.g. genetically induced disorders has attracted a lot of research during the last decades. For instance, *Evolutionary Rule Machine Learning* (ERBML) techniques such as BioHEL [1] or ExSTraCS [8] have been specifically devised to predict protein structures or for the identification and modeling of predictive single nucleotide polymorphisms (SNPs), respectively. Another particular problem domain that falls in the same category is the task of *RNA Secondary Structure Prediction* (RNA-SSP) [12]. RNA is not only an intermediate in the synthesis of proteins from DNA, but also provides essential structural and chemical functions. Since functionality of molecules always depends on their structure, knowing this structure is imperative for understanding and influencing their objective.

In this working paper, we first introduce the state of the art of the *Chemical Reaction Optimisation* (CRO) [3] algorithm, a bio-inspired population-based optimisation heuristic, and describe the pursued domain of application, namely RNA-SSP (Section 2). To this end, the *adaptive CRO* algorithm is enhanced with regard to its optimisation efficiency by means

of reframing it into a parallel implementation which allows for faster convergence towards approximate minima (Section 3). The superior performance is underpinned by results of conducted experiments on well-known and thoroughly selected benchmark functions (Sections 4 and 5). Preliminary results promise improved efficiency in terms of both the number of optimisation steps as well as time consumption. As a second contribution, a way to adequately model the RNA-SSP problem to be solvable by the introduced parallel CRO mechanism (pACRO) will be initially outlined and discussed (Section 6).

II. BACKGROUND

A. Chemical Reaction Optimisation

The chemical reaction optimisation algorithm is a population-based optimisation heuristic which was first devised by Lam and Li in 2010 [3]. It was extended to a real-coded version for solving continuous optimisation problems called RCCRO by introducing a new parameter *stepSize* in [5] and further refined to a so-called *Adaptive CRO* (ACRO) variant, in which the number of predefined (hyper-)parameters is reduced in [10]. ACRO is used as a basis for the proposed pACRO algorithm described later on.

All variants of CRO are built on the same basic structure. The population is composed of *molecules*, which represent possible solutions and can take part in four different kinds of *reactions*. The reactions themselves underly the *laws of energy conservation* and can only occur in accordance to those. All those processes are contained in a closed system, i.e. no energy can be added or lost.

In ACRO, molecules have three necessary attributes as defined in [10]: the *structure* ω , the *potential energy* PE_ω and the *kinetic energy* KE_ω . The structure ω encodes the solution the molecule has reached so far and its specific configuration (encoding) is dependent on the underlying optimisation problem. The potential energy PE_ω denotes the value $f(\omega)$ of the objective function $f : \omega \mapsto \mathbb{R}$ of the molecule's structure, i.e. $PE_\omega = f(\omega)$. The minimisation of PE_ω is the overall aim of the CRO algorithm. The kinetic energy KE_ω serves as a tolerance offset regarding less optimal structures, i.e. PE_ω values, resulting from 'chemical' reactions in order to support escaping from local optima.

The four reaction types, listed in Table I, are similar for all CRO variants: There are two unimolecular reactions, the *on-wall ineffective collision* and the *decomposition*, as well as two bimolecular reactions, the *intermolecular ineffective collision* and the *synthesis* [4].

TABLE I
CRO'S FOUR REACTION TYPES AND THEIR CATEGORISATION INTO SUPPORTING EITHER EXPLORATIVE OR EXPLOITATIVE BEHAVIOUR.

	Reactions	
	<i>unimolecular</i>	<i>bimolecular</i>
exploitation	on-wall ineffective collision	intermolecular ineffective collision
exploration	decomposition	synthesis

All reactions must follow the law of energy conservation. Accordingly, the total amount of energy in the system needs to stay constant at every point in time t . Thus, the condition

$$\sum_{i=1}^{popSize(t)} (PE_{\omega_i}(t) + KE_{\omega_i}(t)) + buffer(t) = C \quad (1)$$

must hold, where $popSize(t)$ denotes the number of molecules at time t , $buffer(t)$ determines the overall amount of energy molecules lose during reactions and C is the value denoting the total energy contained in the system. Additionally, reactions can only take place if an energy threshold is overcome in accordance with the activation energy in real chemical reactions, i.e. the molecule has enough kinetic energy to start a reaction.

1) *On-wall ineffective collision*: The on-wall ineffective collision describes the effects of a molecule colliding with a wall of the container without breaking apart. However, the structure of the molecule changes, i.e. $\omega \rightarrow \omega'$.

In this case, the neighbourhood search operator for the change of ω is realized by *Gaussian perturbation with reflection*. One of the $i = 1 \dots n$ components of the molecule's structure, denoted by $\omega(i)$ is chosen at random and replaced by a new value within the boundaries of the solution space. In RCCRO and ACRO, the perturbation is determined by

$$\omega'(i) = \omega(i) + \Delta_\omega(i), \quad \text{with } \Delta_\omega(i) \sim \mathcal{N}(0, \sigma). \quad (2)$$

In ACRO, the *step size* σ is adaptive according to the 1/5th success rule [10]. This self-adaptation mechanism works as follows: Every n reactions, the number of successful reactions in the last $10n$ reactions is reviewed. If 20 percent of those reactions were successful, the *stepSize* is divided by 0.85, if fewer than 20 percent were successful, *stepSize* is multiplied by 0.85. After generating ω' , the new potential energy $PE_{\omega'}$ is set to $f(\omega')$ and the new kinetic energy $KE_{\omega'} = \alpha(PE_\omega - PE_{\omega'} + KE_\omega)$, with $\alpha \in [KELossRate, 1]$ denoting the factor by which the kinetic energy is reduced in the molecules. The lost kinetic energy is transferred to the buffer, a container to store this lost energy and provide it when necessary.

Before adopting the new values, it is validated if the KE_ω of the molecule is high enough to allow for a reaction, i.e. the energy threshold has been exceeded. This is the case if $PE_\omega + KE_\omega \geq PE_{\omega'}$. If the KE_ω is not high enough, the reaction is aborted and the new values are discarded.

2) *Decomposition*: In a decomposition, the molecule collides with a wall, but breaks apart into two new molecules. In terms of its structure, that means: $\omega \rightarrow \omega'_1 + \omega'_2$

The new structures are generated by a half-total change of the former structure ω . Half of its values are randomly picked and distributed over the new structures. The remaining values of the new structures are randomly picked within the boundaries of the solution space. The new PE values, $PE_{\omega'_1}$

and $PE_{\omega'_2}$, are set to $f(\omega'_1)$ and $f(\omega'_2)$, respectively. The new KE values are computed as follows:

$$E_{dec} = (PE_{\omega} + KE_{\omega} + \delta_1 \cdot \delta_2 \cdot buffer) - (PE_{\omega'_1} + PE_{\omega'_2}) \quad (3)$$

$$KE_{\omega'_1} = E_{dec} \cdot \delta_3 \text{ and } KE_{\omega'_2} = E_{dec}(1 - \delta_3), \quad (4)$$

where all δ_i are random numbers within $[0,1]$.

The decomposition is only valid, if the energy conservation is satisfied, i.e.:

$$(PE_{\omega} + KE_{\omega} + \delta_1 \cdot \delta_2 \cdot buffer) \geq (PE_{\omega'_1} + PE_{\omega'_2}) \quad (5)$$

If so, then the new molecules are instantiated with their respective values, the molecule entering the reaction is removed and the buffer is set to its new value:

$$buffer' = (1 - \delta_1 \delta_2) buffer \quad (6)$$

3) *Intermolecular ineffective collision*: The intermolecular ineffective collision describes the event of two molecules colliding and bouncing apart, similar to the on-wall ineffective collision: $\omega_1 + \omega_2 \rightarrow \omega'_1 + \omega'_2$.

Both molecules experience a change in their structures, following the same procedure as in an on-wall ineffective collision. In both structures, only one value is changed. The PE values are updated by evaluating the objective function on the new structures ω'_1 and ω'_2 . The KE values are updated by

$$KE_{\omega'_1} = E_{inter} \cdot \delta_4 \text{ and } KE_{\omega'_2} = E_{inter} \cdot (1 - \delta_4) \quad (7)$$

$$\text{with } E_{inter} = (PE_{\omega_1} + PE_{\omega_2} + KE_{\omega_1} + KE_{\omega_2}) - (PE_{\omega'_1} + PE_{\omega'_2}). \quad (8)$$

Energy conservation is given if

$$PE_{\omega_1} + PE_{\omega_2} + KE_{\omega_1} + KE_{\omega_2} \geq PE_{\omega'_1} + PE_{\omega'_2}. \quad (9)$$

4) *Synthesis*: A synthesis occurs if two molecules collide and fuse into a single new molecule. In terms of their structures, that means: $\omega_1 + \omega_2 \rightarrow \omega'$.

The new structure ω' is generated using probabilistic select, i.e. choosing its values randomly from ω_1 and ω_2 . $PE_{\omega'}$ is set to $f(\omega')$ and

$$KE_{\omega'} = (PE_{\omega_1} + PE_{\omega_2} + KE_{\omega_1} + KE_{\omega_2}) - (PE_{\omega'}) \quad (10)$$

The energy conservation condition in this case is

$$(PE_{\omega_1} + PE_{\omega_2} + KE_{\omega_1} + KE_{\omega_2}) \geq PE_{\omega'}. \quad (11)$$

5) *Algorithm*: The pseudocode of ACRO is presented in Algorithm 1. For ACRO, only the parameters *popSize*, *molColl* and *changeRate* need to be set. *molColl* describes the ratio between unimolecular and bimolecular reactions. *changeRate* is responsible for the frequency of synthesis and decomposition reactions, in combination with the parameters f_{dec} and f_{syn} for the decomposition and synthesis thresholds. Those thresholds are dependent on the *popSize*, so that the number of molecules does not increase or decrease excessively, i.e. there are about as many syntheses as decompositions.

Algorithm 1 ACRO

```

1: Input: popSize, molColl, changeRate
2: Initialise molecule population M
3: while termination criterion not met do
4:   Adapt  $f_{syn}$  and  $f_{dec}$ 
5:   if Random  $\in [0,1] > molColl$  or  $popSize < 2$  then
6:     Randomly select one molecule
7:     if Random  $\in [0,1] \leq changeRate$  and Random  $\in [0,1] \leq f_{dec}$  then
8:       trigger decomposition
9:     else
10:      trigger on-wall ineffective collision
11:    end if
12:  else
13:    Randomly select two molecules
14:    if Random  $\in [0,1] \leq changeRate$  and Random  $\in [0,1] < f_{syn}$  then
15:      trigger synthesis
16:    else
17:      trigger intermolecular ineffective collision
18:    end if
19:  end if
20:   $\omega^* \leftarrow \min_{\omega} PE_{\omega \in M}$ 
21: end while
22: Output:  $\omega^*$ ,  $PE_{\omega^*}$ 

```

B. RNA Secondary Structure Prediction

The task of RNA-SSP is important as an imperative intermediate step to predict RNA tertiary structures. By knowing the tertiary structure of a molecule, it is possible to gain information on its biological function, and thus, about feasible ways to influence this function, e.g., to provide cures for diseases.

The secondary structure of RNA is provided by the internal bindings in an RNA sequence. RNA consists of four different nucleotide bases capable of forming specific bonds between each other. Those bases are adenine (A), cytosine (C), guanine (G) and uracil (U), with bonds forming between A and U, C and G and G and U. Molecules are usually found in their most stable configuration, i.e. their free energy is minimal. Thus, nucleotide pairs form accordingly.

Most methods for RNA secondary structure prediction are based on a thermodynamic approach. They predict the structure by minimising the free energy of the folded RNA sequence. The necessary information about binding energies of nucleotide pairs and unpaired bases of different substructures is provided by the nearest-neighbour parameter database by Turner et al. [7]

One of the most commonly used algorithms for RNA-SSP is mfold by Zuker et al. [12]. It is based on a dynamic programming approach to minimise the free energy. The downside is, however, the duration of the computation. For long RNA sequences, these methods can be extremely time-consuming. Thus, heuristic approaches to this optimisation task have been

proposed recently. SARNA-Predict, for example, is based on simulated annealing and achieves results as good as those of mfold [6]. Another example is HelixPSO, using particle swarm optimisation to minimise the free energy of RNA secondary structures [2]. To the best of the authors knowledge, there is no study utilising CRO-based approach to accomplish RNA-SSP.

III. PARALLEL ACRO

A. Description of pACRO

A first attempt to parallelise CRO as proposed in [9] was to split the population of molecules between several CPUs, with each CPU running one instance of the conventional CRO algorithm. The buffer is kept in a central coordinating instance which also compares the new molecules obtained by the parallel instances after each step of the algorithm. It then conveys the best molecule back to the parallel instances, which all substitute one of their molecules with the best of the iteration. The idea behind pACRO, however, is to assign an individual thread to each molecule, i.e. making them actors of an AKKA¹ based actor system, with each molecule performing the ACRO algorithm independently. Only essential information should be communicated between the actors. A master actor is responsible for the evaluation of the current optimum and the termination of the system. To prevent the master from an overloaded message queue, two additional actors are implemented, one called helper for pre-evaluating the optimum and the other called market for aiding molecules in bimolecular reactions. The specifics of each of those actors are described in the following:

1) *Molecule*: There are two main differences between molecules in ACRO and its parallel version proposed in this work: (1) Each molecule in pACRO runs its own algorithm. That is no problem for unimolecular reactions, whereas for bimolecular reactions a way of communication was needed. A direct communication between molecules is not desirable, thus, the market was introduced. If one molecule wants to take part in a bimolecular reaction, the market obtains the second molecule and the reaction can take place.

The other difference to the sequential ACRO implementation is that the *stepSize* is no longer a global attribute of the algorithm itself, effective for all molecules. It is now a specific attribute of each single molecule. This was necessary because a repeated update of *stepSize* by the 1/5th success rule has to be communicated between all molecules, as well as the overall reaction count. This would cause delays in the synchronised adaptation. However, we deem an individual step size σ beneficial since this should allow for different exploitation behaviours in different niches within the search space determined by the structures (solutions) encoded by the individual molecules.

The termination criterion for ACRO was the number of function evaluations (FE). To use the same criterion, this number is now distributed among all molecules by dividing the overall number of FE by the number of molecules and setting it

¹<https://akka.io/> [Last accessed: 28.02.2019]

as a parameter for each individual molecule. Thus, the overall number of function evaluations should remain comparable to ACRO.

2) *Master*: The master actor is responsible for the creation of the molecules and the monitoring of the population size (*popSize*). The *popSize* parameter is the only value which is regularly distributed among all molecules. This is done because after a successful decomposition or synthesis, the *popSize* changes. The current *popSize*, however, is necessary for deciding between a decomposition or a synthesis. Thus, that master is informed of those successful reactions and molecules are deleted or added, respectively.

In addition, the master saves the final optimum and is responsible for the termination of the optimisation process after all molecules finished their computation.

3) *Helper*: The only task of the helper is to monitor the development of the optimum. Each time a molecule finds a new minimal structure, it reports the new locally optimal value to the helper, who determines the current overall optimum. This task has to be distributed to an additional actor to prevent the master from a congested mailbox, which would delay the updates of *popSize*.

4) *Market*: Since molecules should not share their information in an uncoordinated manner, the market was introduced to mediate bimolecular reactions. If a molecule wants to take part in a bimolecular reaction, it reports its intention to the market. As soon as the market has received two requests, it provides those two molecules with the necessary information. This corresponds closely with the random selection of two molecules in ACRO.

This design allows for any number of molecules to receive their own thread, thus running the algorithm in parallel and rendering it highly scalable as will be shown in Table III. It has to be noted, however, that there are three additional threads for master, helper and market necessary for monitoring and coordination purposes.

IV. EXPERIMENTAL SETTING

The pACRO algorithm is tested on a subset of those benchmark functions used for testing RCCRO and ACRO (cf. [5], [10]). The selected functions and their specifics are shown in Table II.² Parameters were also set according to the propositions for ACRO: the number of function evaluations was set to 300'000, *molColl* to 0.2 and *changeRate* to 10^{-4} . For further examination of the parameter sensitivity of pACRO, three additional tests were performed with a change in exactly one of those parameters, while leaving the remaining unchanged: in one test the number of function evaluations was set to 600'000, in the next, *molColl* was set to 0.5 and in the last case, *changeRate* was set to 10^{-3} .

²The benchmark functions were chosen to give an adequate review on the functioning of pACRO. Those taken from [5] are named accordingly and are either functions, where RCCRO performed well, or badly. f_9 was added to see how pACRO performs with many local minima. Functions taken from [10] were named according to the paper with the letter 'A' marking their utilisation for the ACRO algorithm. They were chosen to provide shifted versions of benchmark functions from [5].

TABLE II

CHARACTERISTICS IN TERMS OF DEFINITION, DIMENSIONALITY, DOMAIN AND GLOBAL MINIMUM OF THE BENCHMARK FUNCTIONS USED TO COMPARE ACRO AND RCCRO WITH THE NOVEL PACRO.

Function	Name	n	Search space S	f_{min}
$f_{A1}(x) = \sum_{i=1}^n z_i^2 *$	Shifted sphere function	30	$[-100, 100]^n$	0
$f_{A11}(x) = -20 \exp(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^n z_i^2}) - \exp(\frac{1}{n} \sum_{i=1}^n \cos 2\pi z_i) + 20 + e *$	Shifted Ackley's function	30	$[-100, 100]^n$	0
$f_1(x) = \sum_{i=1}^n x_i^2$	Sphere Model	30	$[-100, 100]^n$	0
$f_2(x) = \sum_{i=1}^n x_i + \prod_{i=1}^n x_i $	Schwefel's problem 2.22	30	$[-10, 10]^n$	0
$f_5(x) = \sum_{i=1}^{n-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	Generalised Rosenbrock's function	30	$[-30, 30]^n$	0
$f_9(x) = \sum_{i=1}^n (x_i^2 - 10 \cos(2\pi x_i) + 10)$	Generalised Rastrigin's function	30	$[-5.12, 5.12]^n$	0
$f_{10}(x) = -20 \exp(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2}) - \exp(\frac{1}{n} \sum_{i=1}^n \cos 2\pi x_i) + 20 + e$	Ackley's function	30	$[-32, 32]^n$	0

* $z = x - \sigma$ with σ being a shifting vector

* $z = (x - \sigma) \times 0.32$ with σ being a shifting vector

The optimal population size determined for ACRO was 20 [10]. In pACRO, however, the variations in the results for different numbers of molecules are relevant, e.g. changes in computation time and the precision of the predicted optimum. Therefore, for each parameter setting the algorithm was run with 2, 4, 8, 16, 20, 32 and 64 molecules. The standard number of FE was set to 300'000, which were distributed over the molecule, i.e. in the case of 2 molecules, each performs 150'000 FE.

All experiments were performed on an Intel(R) Xeon(R) CPU E5-2640 0 @ 2.50GHz CPU based multi-core computing server with 48 GB DDR3 RAM and Linux operating system with Kernel version 4.14.92 with exclusive use, i.e. no other computing intensive processes had to be scheduled by the OS during the computation.

V. RESULTS

This section summarises the simulation results and statistics of pACRO as described in Section 4 and compares them to the results of RCCRO and ACRO as reported in [5] and [10], respectively.

A. Results of pACRO

Table III shows the results for each selected benchmark function. It provides the mean optimum value, its standard deviation, the median optimum value and the average computing time for 50 simulation runs, respectively. 300'000 function evaluations (FE) denotes simulations with the standard parameter configuration of ACRO, other parameter settings describe the parameter changed. Respective results of RCCRO and ACRO are given at the bottom of Table III.

All simulations display the best results for only 2 molecules. With increasing number of molecules, the optimum value also increases, while computation time decreases. Different parameter settings for f_{A1} show changes in the optimum value for higher molecule numbers. As expected, increasing

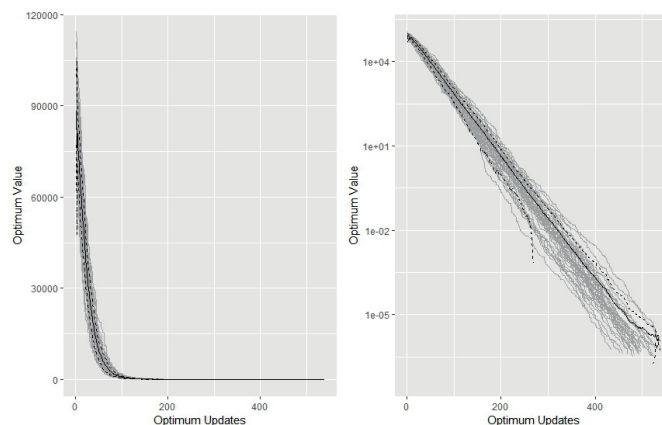


Fig. 1. **Development of Optimum** over the optimum updates for 2 molecules and benchmark function f_{A1} with standard 300'000 FE. The grey lines show the optimum development of all 50 runs, the black line shows the mean optimum development with the standard deviation (dotted black line). In the left picture, a linear scale is applied, in the right one the scale is logarithmic.

the number of function evaluations provides a better optimum value. A higher value for *molColl* worsens the optimum value, while a higher value of *changeRate* has almost no effect. For f_9 a higher *changeRate* provides a better mean optimum value for smaller numbers of molecules.

In comparison to the results of RCCRO and ACRO, the mean optimum values of pACRO in the simulation runs with two molecules are similar. For example, the mean optimum value for f_{A1} and f_1 provided by pACRO is worse than the given values, for f_{10} pACRO presents a better result.

In addition, the development of the optima was analysed. Figure 1 shows the evolution of the optimum value over its updates for all 50 simulation runs of f_{A1} with 300'000 FE. Over the simulation, the optimum values are decreasing exponentially.

The plots in Figure 2 display the achieved optimum value

TABLE III
RESULTS OF PACRO

# Mols	Test Functions		f_{A1}				f_{A11}				f_1		f_2		f_5		f_9		f_{10}			
	Parameter Setting	FE	600'000 FE	molColl = 0.5	changeRate = 0.001	300'000 FE	300'000 FE	300'000 FE	300'000 FE	300'000 FE	300'000 FE	300'000 FE	300'000 FE	300'000 FE	300'000 FE	300'000 FE	300'000 FE	300'000 FE	300'000 FE	300'000 FE	300'000 FE	
2	mean	0.00064112	0.00021033	0.00020722	0.00103137	0.00017637	0.00055156	0.00232619	0.00232619	0.00232619	0.00232619	0.00232619	0.00232619	0.00232619	0.00232619	0.00232619	0.00232619	0.00232619	0.00232619	0.00232619	0.00232619	
	stdev	0.00218341	0.00077447	0.00114001	0.00709282	2.3507e-05	0.0258557	0.01023039	0.01023039	0.01023039	0.01023039	0.01023039	0.01023039	0.01023039	0.01023039	0.01023039	0.01023039	0.01023039	0.01023039	0.01023039	0.01023039	
	median	7.4214e-07	1.6575e-07	6.0164e-07	7.4003e-07	0.00017580	6.2818e-07	0.00028724	0.00028724	0.00028724	0.00028724	0.00028724	0.00028724	0.00028724	0.00028724	0.00028724	0.00028724	0.00028724	0.00028724	0.00028724	0.00028724	0.00028724
4	avg time	62 s	106.3 s	60.3 s	64 s	95.7 s	59.3 s	71.3 s	71.3 s	71.3 s	71.3 s	71.3 s	71.3 s	71.3 s	71.3 s	71.3 s	71.3 s	71.3 s	71.3 s	71.3 s	71.3 s	71.3 s
	mean	0.02996092	0.01048149	0.25238	0.03930199	0.03024216	0.02141783	0.04758404	0.04758404	0.04758404	0.04758404	0.04758404	0.04758404	0.04758404	0.04758404	0.04758404	0.04758404	0.04758404	0.04758404	0.04758404	0.04758404	0.04758404
	stdev	0.05204997	0.01960795	0.250286	0.09740716	0.02262285	0.0273141	0.04244306	0.04244306	0.04244306	0.04244306	0.04244306	0.04244306	0.04244306	0.04244306	0.04244306	0.04244306	0.04244306	0.04244306	0.04244306	0.04244306	0.04244306
8	median	0.00583298	0.00156617	0.2316787	0.00273251	0.0220382	0.00566861	0.03263476	0.03263476	0.03263476	0.03263476	0.03263476	0.03263476	0.03263476	0.03263476	0.03263476	0.03263476	0.03263476	0.03263476	0.03263476	0.03263476	0.03263476
	avg time	45.7 s	83.7 s	70.3 s	48 s	103.7 s	50.3 s	47.3 s	47.3 s	47.3 s	47.3 s	47.3 s	47.3 s	47.3 s	47.3 s	47.3 s	47.3 s	47.3 s	47.3 s	47.3 s	47.3 s	47.3 s
	mean	0.2785354	0.0729979	1.05302	0.4910296	0.123196	0.2461529	0.1543363	0.1543363	0.1543363	0.1543363	0.1543363	0.1543363	0.1543363	0.1543363	0.1543363	0.1543363	0.1543363	0.1543363	0.1543363	0.1543363	0.1543363
16	stdev	0.2703504	0.06419369	1.176745	0.6360837	0.02654938	0.1586102	0.04709884	0.04709884	0.04709884	0.04709884	0.04709884	0.04709884	0.04709884	0.04709884	0.04709884	0.04709884	0.04709884	0.04709884	0.04709884	0.04709884	0.04709884
	median	0.1774678	0.04691845	0.6848272	0.208632	0.124223	0.18821	0.143481	0.143481	0.143481	0.143481	0.143481	0.143481	0.143481	0.143481	0.143481	0.143481	0.143481	0.143481	0.143481	0.143481	0.143481
	avg time	30.7 s	53.7 s	35.7 s	30.3 s	38.7 s	32 s	35.3 s	35.3 s	35.3 s	35.3 s	35.3 s	35.3 s	35.3 s	35.3 s	35.3 s	35.3 s	35.3 s	35.3 s	35.3 s	35.3 s	35.3 s
20	mean	2.537692	0.728842	5.110564	3.011173	0.5320096	2.780803	0.5341266	0.5341266	0.5341266	0.5341266	0.5341266	0.5341266	0.5341266	0.5341266	0.5341266	0.5341266	0.5341266	0.5341266	0.5341266	0.5341266	0.5341266
	stdev	0.7312576	0.2860695	2.635963	1.542178	0.0665449	1.033008	0.08460809	0.08460809	0.08460809	0.08460809	0.08460809	0.08460809	0.08460809	0.08460809	0.08460809	0.08460809	0.08460809	0.08460809	0.08460809	0.08460809	0.08460809
	median	2.475817	0.654857	4.706416	2.588527	0.5210552	2.737537	0.5265556	0.5265556	0.5265556	0.5265556	0.5265556	0.5265556	0.5265556	0.5265556	0.5265556	0.5265556	0.5265556	0.5265556	0.5265556	0.5265556	0.5265556
32	avg time	28 s	44.3 s	37.7 s	32 s	33.7 s	28 s	33 s	33 s	33 s	33 s	33 s	33 s	33 s	33 s	33 s	33 s	33 s	33 s	33 s	33 s	33 s
	mean	4.551799	1.238949	9.326067	5.088078	0.8157004	4.427599	0.7237729	0.7237729	0.7237729	0.7237729	0.7237729	0.7237729	0.7237729	0.7237729	0.7237729	0.7237729	0.7237729	0.7237729	0.7237729	0.7237729	0.7237729
	stdev	1.39805	0.4814507	5.865416	1.862541	0.1059303	1.583828	0.1137008	0.1137008	0.1137008	0.1137008	0.1137008	0.1137008	0.1137008	0.1137008	0.1137008	0.1137008	0.1137008	0.1137008	0.1137008	0.1137008	0.1137008
64	median	4.427456	1.141839	8.638972	4.971818	0.8137083	4.161308	0.7287443	0.7287443	0.7287443	0.7287443	0.7287443	0.7287443	0.7287443	0.7287443	0.7287443	0.7287443	0.7287443	0.7287443	0.7287443	0.7287443	0.7287443
	avg time	27.3 s	43.7 s	35 s	31.6 s	32.7 s	27 s	31.3 s	31.3 s	31.3 s	31.3 s	31.3 s	31.3 s	31.3 s	31.3 s	31.3 s	31.3 s	31.3 s	31.3 s	31.3 s	31.3 s	31.3 s
	mean	14.20874	4.016768	25.90758	15.44058	1.485056	15.01212	1.19478	1.19478	1.19478	1.19478	1.19478	1.19478	1.19478	1.19478	1.19478	1.19478	1.19478	1.19478	1.19478	1.19478	1.19478
RCCRO	stdev	4.451612	1.549302	14.51822	4.726664	0.1786753	4.606871	0.1443033	0.1443033	0.1443033	0.1443033	0.1443033	0.1443033	0.1443033	0.1443033	0.1443033	0.1443033	0.1443033	0.1443033	0.1443033	0.1443033	0.1443033
	median	13.59496	3.699871	23.79177	14.90183	1.517003	14.17463	1.180215	1.180215	1.180215	1.180215	1.180215	1.180215	1.180215	1.180215	1.180215	1.180215	1.180215	1.180215	1.180215	1.180215	1.180215
	avg time	25.7 s	41.7 s	32.7 s	33.3 s	36 s	26.3 s	29 s	29 s	29 s	29 s	29 s	29 s	29 s	29 s	29 s	29 s	29 s	29 s	29 s	29 s	29 s
ACRO *	mean	50.68482	16.35394	116.389	55.84952	2.709483	51.24243	2.319055	2.319055	2.319055	2.319055	2.319055	2.319055	2.319055	2.319055	2.319055	2.319055	2.319055	2.319055	2.319055	2.319055	2.319055
	stdev	10.36926	4.619848	47.05	16.63377	0.1925252	11.87245	0.2475103	0.2475103	0.2475103	0.2475103	0.2475103	0.2475103	0.2475103	0.2475103	0.2475103	0.2475103	0.2475103	0.2475103	0.2475103	0.2475103	0.2475103
	median	50.42528	15.97653	117.0399	52.49849	2.734081	51.86821	2.319325	2.319325	2.319325	2.319325	2.319325	2.319325	2.319325	2.319325	2.319325	2.319325	2.319325	2.319325	2.319325	2.319325	2.319325
ACRO *	avg time	23 s	38.7 s	31.7 s	33.6 s	40.7 s	25.3 s	27.7 s	27.7 s	27.7 s	27.7 s	27.7 s	27.7 s	27.7 s	27.7 s	27.7 s	27.7 s	27.7 s	27.7 s	27.7 s	27.7 s	27.7 s
	mean	0.0000e+00	8.4478e-02	6.427e-07	2.099e-07	2.196e-03	2.706e+01	2.706e+01	2.706e+01	2.706e+01	2.706e+01	2.706e+01	2.706e+01	2.706e+01	2.706e+01	2.706e+01	2.706e+01	2.706e+01	2.706e+01	2.706e+01	2.706e+01	2.706e+01
ACRO *	stdev		4.341e-04	3.427e+01	3.427e+01	4.341e-04	3.427e+01	3.427e+01	3.427e+01	3.427e+01	3.427e+01	3.427e+01	3.427e+01	3.427e+01	3.427e+01	3.427e+01	3.427e+01	3.427e+01	3.427e+01	3.427e+01	3.427e+01	3.427e+01

* Values are taken from [5], [10] and need to be reviewed in an own implementation; for ACRO: all values smaller than 1E-08 were assumed 0

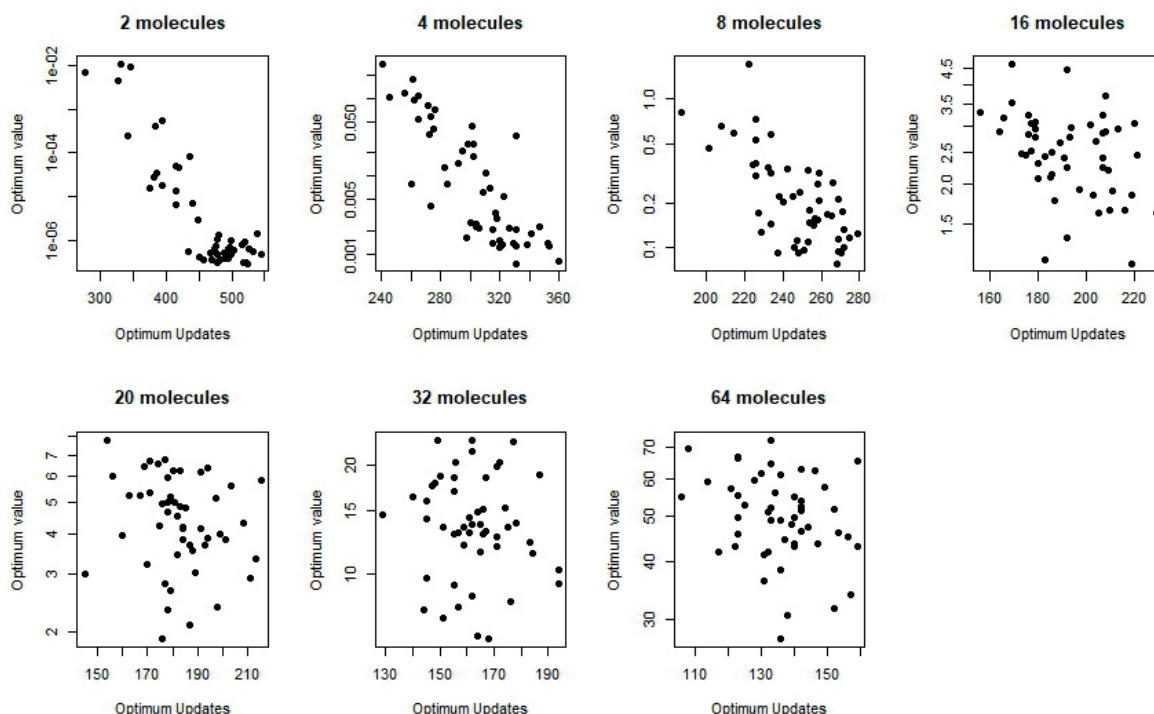


Fig. 2. **Distribution of Optimum Values** for 2, 4, 8, 16, 20, 32 and 64 molecules. Each plot shows the final optimum values and the corresponding number of updates for 50 runs.

and its corresponding number of updates, i.e. the number of better optimum values being found in 300'000 FE, for the test function f_{A1} . For 2 molecules, most optimum values are updated about 500 times in one simulation run and their value reaches about $1E - 07$. There are, however, some aberrant values up to $1E - 02$ with a much smaller number of updates. For a higher population size, the overall number of optimum updates decreases, while the optimum value increases, and the optimum values differ more strongly in one population size.

B. Discussion

Overall, pACRO shows similar results to former CRO variants. However, this is only if a population size of two molecules is chosen.³ For higher population sizes, the pitch between the expected and the resulting optimum value increases, a problem which still needs to be addressed. A further examination of the increase in the optimum value with an increase in population size is shown in Figure 2. The computation times, however, decrease with an increase in molecules. Changes in the parameter setting of the pACRO algorithm also provided the expected results. A higher number of function evaluation leads to an improvement of the optimum value, especially for increasing population sizes. Increasing the *molColl*, i.e. the ratio of unimolecular to bimolecular reactions, has a negative effect on the resulting optimum. An increase of the *changeRate*, i.e. more synthesis and decomposition

³Unfortunately, the original CRO source codes are no longer available to be compared to the pACRO implementation.

reactions, has almost no effect on the results, except in the case of benchmark function f_9 , which includes many local optima. For this function, more exploratory reactions prove to be advantageous, as was expected.

pACRO is shown to provide a promising tool to solve many kinds of optimisation problems. It performs well on the selected benchmark functions. The next step is its adaptation to the more complex problem of RNA-SSP.

VI. TOWARDS RNA-SSP WITH PACRO

The thermodynamic approach, i.e. minimising the free energy, makes RNA secondary structure prediction a suitable optimisation problem for pACRO. To adjust the problem to the algorithm and vice versa, two main subjects need to be addressed: The solution representation of the algorithm must be fitted to RNA-SSP and the four reactions changing it need to be adapted.

A. Solution representation

The solution, i.e. the structures of the molecules in pACRO, will be represented by a set of helices, similar to the approach used in SARNAPredict [6]. Helices are stretches of the RNA sequence where binding between nucleotides is possible. The generation of this set, however, differs from previous approaches. Instead of iterating over the RNA sequence, the sequence is used to create a matrix containing information of all possible nucleotide pairs. Then, a particle based algorithm is used to search this matrix for stretches of nucleotide pairs

with a minimal length. This particle approach allows for not only one particle to search for helices, but can be extended to any number of particles. It is also easy to parallelise if time-consumption is too high.

The particle generates a set of all possible helices the RNA sequence is able to form. From this set, the pACRO structures are formed by drawing random subsets. These are validated, i.e. helices not compatible with each other are removed, as described for SARNA-Predict [6].

These subsets represented in the structures allow evaluation in terms of their free energy, according to the nearest-neighbour model [7]. They can also be changed in the four reactions provided by pACRO.

B. Reactions

The ACRO algorithm is designed to incorporate a molecular structure containing numerical values. Thus, the four reaction types provide only possibilities to change those values. In this approach to RNA-SSP, however, the structure will consist of helix objects, each defined by a sequence and a starting point. Therefore, it is necessary to adapt the reactions of the pACRO algorithm to the structures given, but without disturbing the function of the algorithm.

For synthesis and decomposition reactions, there will be only minor changes in the implementation. With a structure composed of helices, it is still possible to perform probabilistic select and half-total change. It is, however, necessary to validate the structures after the reaction to prevent it from containing mutually exclusive helices. In both cases it might additionally be required to inhibit the structures from growing too short due to validation.

The on-wall ineffective collision and the intermolecular ineffective collision need more alteration. In both, only one numerical component of the structure is changed by multiplication with a value generated by Gaussian perturbation with reflection depending on the *stepSize*. For a structure composed of helices, this procedure is not feasible. One possible way to solve this problem is to substitute single helices with more or less similar ones. Similarity between helices can be defined in regard of their starting points and sequences. Thus, when the value of *stepSize* is high, i.e. the value in the structure is changed strongly, the helix is substituted by another with low similarity. Accordingly, for a lower *stepSize*, a more similar helix is included.

Implementing these changes is one possible way to adapt the pACRO algorithm to the problem of RNA-SSP. The feasibility of this approach still needs to be verified and changes due to unforeseen difficulties might need to be made.

VII. CONCLUSION

In the present paper, a novel means for a parallel version of the Chemical Reaction Optimisation (CRO) heuristic has been developed in view of the utilisation in Intelligent RNA Prediction Systems. The presented pACRO algorithm yields superior optimisation efficiency in terms of steps to approximate optima (so far only for the 2 molecule configuration) as

well as reduced time consumption due to a speedup obtained by utilising AKKA-based parallelisation. Accordingly, the algorithm's scalability is substantially improved, allowing for the application on more complex search spaces in contrast to its sequential version. Regarding application of pACRO within Intelligent RNA Prediction Systems, the underlying optimisation task that needs to be solved in the context of RNA-SSP has been outlined and an adequate solution candidate encoding as well as a novel methodology to helix identification have been briefly sketched.

At the time of preparing this working paper, preliminary experiments already indicated positive results when the introduced pACRO algorithm is applied to the RNA-SSP optimisation problem. However, a more thorough assessment of the preliminary results as well as comprehensive comparative studies with further state of the art optimisation heuristics have to be conducted which is already part of current research activities. Further aspects that are going to be investigated in the near future comprise: (i) the refinement of the exploration mechanisms of CRO, (ii) the incorporation of information obtained from neighbouring molecules in a PSO-resembling fashion to further control the optimisation process as proposed in [11], and, (iii) the extension of pACRO toward a more self-adaptive optimisation technique in order to reduce the influence of a priori, possibly inappropriately set hyperparameters on the optimisation progress.

REFERENCES

- [1] Jaume Bacardit, Edmund K. Burke, and Natalio Krasnogor. Improving the scalability of rule-based evolutionary learning. *Memetic Computing*, 1(1):55–67, Mar 2009.
- [2] Michael Geis and Martin Middendorf. Particle swarm optimization for finding RNA secondary structures. *International Journal of Intelligent Computing and Cybernetics*, 4(2):160–186, 2011.
- [3] Albert Y. S. Lam and Victor O. K. Li. Chemical-reaction-inspired metaheuristic for optimization. *IEEE Transactions on Evolutionary Computation*, 14(3):381–399, 2010.
- [4] Albert Y. S. Lam and Victor O. K. Li. Chemical reaction optimization: a tutorial. *Memetic Computing*, 4(1):3–17, 2012.
- [5] Albert Y. S. Lam, Victor O. K. Li, and James J. Q. Yu. Real-coded chemical reaction optimization. *IEEE Transactions on Evolutionary Computation*, 16(3):339–353, 2012.
- [6] Herbert H. Tsang and Kay C. Wiese. Sarna-predict: Accuracy improvement of rna secondary structure prediction using permutation-based simulated annealing. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 7(4):727–740, 2010.
- [7] Douglas H. Turner and David H. Mathews. NNDB: the nearest neighbor parameter database for predicting stability of nucleic acid secondary structure. *Nucleic Acids Research*, 38(suppl_1):280–282, 2009.
- [8] Ryan J. Urbanowicz and Jason H. Moore. Exstracs 2.0: description and evaluation of a scalable learning classifier system. *Evolutionary Intelligence*, 8(2):89–116, 2015.
- [9] Jin Xu, Albert Y. S. Lam, and Victor O. K. Li. Parallel chemical reaction optimization for the quadratic assignment problem. In *The 2010 World Congress in Computer Science, Computer Engineering, and Applied Computing (Worldcomp 2010)*, Las Vegas, NV, pages 125–131, 2010.
- [10] James J. Q. Yu, Albert Y. S. Lam, and Victor O. K. Li. Adaptive chemical reaction optimization for global numerical optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 3192–3199. IEEE, 2015.
- [11] Min Zhang, Liang Chen, and Xin Chen. An advanced chemical reaction optimization algorithm based on balanced local and global search. *Mathematical Problems in Engineering*, 2018:1–16, 2018.
- [12] Michael Zuker and David Sankoff. Rna secondary structures and their prediction. *Bulletin of Mathematical Biology*, 46(4):591–621, 1984.