This may be the author's version of a work that was submitted/accepted for publication in the following source:

Wee, Chee & Nayak, Richi (2020) Adaptive Data Replication Optimization Based on Reinforcement Learning. In Proceedings of the 2020 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE, United States of America, pp. 1210-1217.

This file was downloaded from: https://eprints.qut.edu.au/206925/

© Consult author(s) regarding copyright matters

This work is covered by copyright. Unless the document is being made available under a Creative Commons Licence, you must assume that re-use is limited to personal use and that permission from the copyright owner must be obtained for all other uses. If the document is available under a Creative Commons License (or other specified license) then refer to the Licence for details of permitted re-use. It is a condition of access that users recognise and abide by the legal requirements associated with these rights. If you believe that this work infringes copyright please provide details by email to qut.copyright@qut.edu.au

License: Creative Commons: Attribution-Noncommercial 4.0

Notice: Please note that this document may not be the Version of Record (i.e. published version) of the work. Author manuscript versions (as Submitted for peer review or as Accepted for publication after peer review) can be identified by an absence of publisher branding and/or typeset appearance. If there is any doubt, please refer to the published source.

https://doi.org/10.1109/SSCI47803.2020.9308306
Abstract - Data replication plays an important role in enterprise IT landscapes, where data is shared among multiple IT systems. IT administrators need to tune the replicating software’s configuration setting for it to perform at its optimum level. It is a challenge to continue optimizing the software’s configuration to keep up with the fluctuating workload in a dynamic business environment. We propose a novel approach of using reinforcement learning with meta-heuristics to create an adaptive optimization method for data replication software. The experimental results show the replicating software managed by the proposed approach can perform at an optimum level despite consistently working under changing workloads.

Keywords - Data replication, optimization, meta-heuristics, deep reinforcement learning

I. INTRODUCTION

Enterprise Data Replication (DRE) is an important function across numerous organizations where information is replicated and shared among multiple IT systems to support various business needs. It also serves as a form of data security and high availability should any mission-critical system fail catastrophically, or distribute the workload across multiple sites for remote processing [1]. The software that supports this form of high-speed data replication is grouped into two categories: 1) High-speed transfer with no alteration to the data or; 2) Slow transfer but with extensive data transformation via Extract-Transform-Load (ETL) process [2]. The software that supports these two categories are distinctively different from one another. There are several types of software that are prevalent in the industry that supports the former category’s data transfer. The majority of the software follows a common set up which comprises both operating parameters and configuration settings [2]. The parameters control the characteristics of the software while the configuration setting controls the replicating tables which include the replicating routes and the constraints that are associated with them [3]. Besides this, other factors can impact the replicating software’s overall throughput performance such as the tables’ attributes and contents as well as the volumes, velocity, and variety of data changes created by different IT systems [3, 4].

Another challenge faced in the DRE is the ever-changing landscape of the IT corporate structure, where the IT systems constantly change in conjunction with the dynamism of the business environment. In any given period, there will be variation in the business functions that are made against the DBs such as data bulk loading, large batch processing, month-end reporting, year-end consolidation, transactional processing, or data transformation. The activities can either occur on an ad-hoc basis or repeat in a cycle. Each set of business IT system’s activities creates a distinct set of works against the IT systems this needs changing the data as required. This, in turn, must be replicated to the other IT systems that depend on it. It is important to keep the operation of data replication at the peak level to maintain a high quality of service level and customer satisfaction. However, the optimization for the DRE’s configuration enables the replication process to perform at its best for a point in time, it may not be relevant in other periods as the workload or business’s operation could have evolved. Therefore, the effort to tune the DRE must be repeated continually to keep its operation at peak condition.

We propose a novel approach that is based on the model-free reinforcement learning in conjunction with the metaheuristics to create an adaptive agent that can optimize the DRE’s replicating software configuration continually for a dynamic changing IT environment [5]. To our best of knowledge, there is no prior research in the field of optimizing the replicating tool’s configuration using deep reinforcement learning while there are other methods that propose the use of heuristic and statistical models. These traditional methods fail to produce the most optimized results for a large period. The computational search space to find the best configuration setting for the replicating tables’ arrangement is high, and the difficulty to adjust the configuration setting continually is high with the multiple variances of workloads in a period. Experimental results indicate that an agent behavior is sensitive to the changes in workload environment and it uses the best configuration setting on the DRE to keep the replication process operating at its peak level. This paper is organized as follows. Section 2 shows the related works and background, section 3 presents the adaptive optimizing agent as well as the participating components and knowledge that contribute to it. Section 4 devotes to the analysis and experiment results. The last section covers the discussion and gives the conclusion.

II. RELATED WORKS

Data replication is used to share information across multiple systems that support different business functions [1, 6]. There exist two main strategies in data replication deployment based on time and space [7]. Each strategy requires a detailed analysis of the requirement for the data replication such as data’s attributes, the volume of data that needs to be transferred including the data transport’s complexity, and the throughput and turnaround time of the changes [7]. Several evaluation metrics have been proposed to measure each replication strategy’s effectiveness [7]. A set of measurements has also been proposed that focus on the execution time and the bandwidth usage that lead toward an optimum configuration design [8].
Both of these measures stress a high emphasis on measurement and controls. The use of a centralized dynamic scheduling and replica placement strategy has been proposed to control both the data and tasking schedule in the replication process [9].

The replication optimization can be considered as a combinatorial optimization problem (COP), which is an NP-hard problem where no optimal solution can be discovered within acceptable timing and resources [10]. Replication optimization is best obtained through metaheuristics. This type of method is commonly used in both academia and industry and they can be generally grouped into population-based and single solution-based [10]. Algorithms such as Tabu-Search or Simulated Annealing (SA) are popular algorithms [10]. Other metaheuristics such as genetic algorithm, ant colony, or particle swarm optimization have more capabilities for large problem space and have a huge population of solutions [10]. However, in this research, the requirement is to continuously interact with the DRE which is constantly changing and to support a dynamic working environment. Even if the replication software has been optimized for a period, it will not suffice as the business IT workload keeps changing throughout. Therefore, this requires regular recalibration and adjustment to keep the replicating software at its peak level at all times.

Designing algorithms to solve combinatorial problems is both laborious and time-consuming which requires numerous cycles of discovering and validating solutions. In this paper, we propose to deploy an agent that can use what it has learned from a series of optimization processes and can act in response to new changes in the DRE. This is done by activating configurations from its learned knowledge-base. This way, the agent can recalibrate the DRE adaptively and effectively, thereby cutting down time wastage and reducing the unnecessary cycle of works [11]. The model-based reinforcement learning that is represented by the Markov decision process and the guided-policy search uses existing optimization methods to search for the best actions for the problem's different states [12]. Table 1 lists existing methods of optimizing DRE along with their rating according to complexity and effort of implementation. The rating is based on the feedback from experienced IT administrators from the industry using the Delphi method [13].

The basic setup, that comes default with the installation, is the simplest but most inefficient form of configuring the system where only one queue is used. There is no planning or strategizing the tables to be replicated in the configuration setting, but rather every table is assigned to one single queue. It may be generally acceptable if the number of tables is small and the volume of data changes is low, but the replication performance degrades exponentially when the tables’ load increases [14]. In the real world, a set of best practices, that have been published by software vendors, is employed to optimize the DRE. The guidelines generally cater to all forms of replication scenarios, and they recommend to share the replication workload by splitting the list of tables across several queues. However, this type of practice does not consider the attributes of the tables or the characteristics of the data change volumes. There is a high chance of having tables with either large volumes or high frequency of data changes congregating together. Thereby, it increases the chance of clogging a common queue [15]. Another practice is to balance best practices with industrial experiences. IT administrators who have worked with data replications understand the characteristics of the tables in replications and re-arrange tables among the queues following their data changing activities. There are two approaches here - the first one is to divide the number of tables evenly across a higher number of queues whereas the other method is to group those with a similar volume of activities together across a few numbers of queues [15].

These approaches are challenged because of the dynamic changing environment of the DRE. On the other hand, data science practitioners also struggle to keep constantly retraining their models to just keep up. Several researchers have proposed systems that can be highly adaptive to the changes and be autonomous enough to recalibrate their models with the minimum human intervention [16]. In the last few years, Deep Reinforcement Learning (DRL) [17] has been used by researchers to develop intuitive and adaptive machine learning-based applications such as adaptive forecasting of load demand for the electricity utilities [18], and anomaly detection for database systems [19]. These novel systems use ensemble methods of DRL and other machine learning algorithms to serve specific needs that other sole machine learning algorithms will need more resources and time to compute [17, 19]. Similar to these works, in this paper, we propose a novel approach to combine DRL with metaheuristic to optimize data replication setup against a fast-changing IT environment.

### Table 1 – Data Replication’s Configuration Optimizing Techniques (LOW 1 TO HIGH 10)

| Method | Complexity | Effort | Remark |
|--------|------------|--------|--------|
| Out-of-box setup [14] | 1 | 1 | Requires no knowledge. Passive effort. All tables are assigned to one single queue. Worst replication throughput of all. Unscalable. Most undesirable form of deployment. |
| Manual [15] | 1 | 2 | Only need basic knowledge and skill. Passive effort with no due consideration to the table’s activities. Poor replication throughput. Not scalable. Need regular tuning. |
| Best practice, decision rule-based [15] | 4 | 3 | Require medium level skill. Taking into tables’ size and volumes. Medium replication throughput. Not scalable. Need regular tuning. |
| Economic Model-based [20] | 7 | 6 | Require a medium level of knowledge and skill. Balancing the queues based on tables’ changes activities and size. Medium-high replication throughput. Not scalable. Need regular tuning. |
| Statistical, metaheuristic-based [21] | 5 | 7 | Require a higher level of knowledge and skill. Tune by metaheuristics against tables’ changes activities and size. High replication throughput. Limited scalability. Need regular tuning. Not adaptive |

**III. Adaptive Data Replication Optimization (ADRO)**

The proposed Adaptive Data Replication Optimization (ADRO) agent uses the ensemble methods of Reinforcement Learning (RL) and metaheuristic optimization (MO) in its two main modules. The first module uses RL to interact with the Data Replication Environment (DRE) by receiving its state’s and respond with the action of applying changes to the configuration setting on the DRE’s replicating system. The second module comprises of the optimizing routine where the
agent uses the metaheuristic method to find the best configuration setting for the DRE’s replicating software for a given workload at a specific time.

**Fig. 2 - Shareplex’s data replication architecture**

Figure 1 shows an overview of ADRO and its functions as follows. The IT administrator sets a specific interval in which the ADRO agent needs to interact and change the configuration settings for a period, preferably a day or a week depending on the rate of workload changes within the systems. DB replay [22] is used to capture the workload in the Production DBs and the captured replay files are then copied over to the test environment where the ADRO agent can perform regressive testing without harming the Production environment.

The test environment is a duplicate of the production systems with identical hardware and software setup. The copied DBReplay’s files [22] represent the state of the DRE’s source DB for that specific period and they are used by the agent’s MO module to find the optimum configuration setting in the test environment. In the MO module, the optimizing algorithm enters into a highly iterative state where it goes through these steps: Firstly, it resets the test environment to a baseline, where it applies a randomly generated configuration setting to the replicating software. This is followed by running the DB-Replay files which recreate the workloads in the source DBs. Next, it monitors and records the timings for the DRE to complete the replication of the data changes to the target. This forms one cycle for a single configuration’s creation, testing, and measurement.

**TABLE 2 – SHAREPLEX’S CONFIGURATION FILE**

| Source Table | Destination Table | Routing Map |
|--------------|-------------------|-------------|
| HR.COUNTRIES | HR.COUNTRIES      | src_svr:Q0*tgt_svr(o.tgt_db) |
| HR.DEPARTMENTS | HR.DEPARTMENTS | src_svr:Q0*tgt_svr(o.tgt_db) |
| HR.EMPLOYEES  | HR.EMPLOYEES      | src_svr:Q1*tgt_svr(o.tgt_db) |
| HR.JOBS       | HR.JOBS           | src_svr:Q2*tgt_svr(o.tgt_db) |
| HR.JOB.HISTORY | HR.JOB.HISTORY   | src_svr:Q2*tgt_svr(o.tgt_db) |
| HR.LOCATIONS  | HR.LOCATIONS      | src_svr:Q2*tgt_svr(o.tgt_db) |
| HR.REGIONS    | HR.REGIONS        | src_svr:Q2*tgt_svr(o.tgt_db) |

The MO module interacts under the Simulated Annealing (SA) algorithm, running hundreds of such configurations’ test cycles until the best configuration is found under a finite loop. It is passed into the agent’s RL module then decides based on its DRL logic. It normalizes the cost of the configuration and stores the details into its knowledgebase. The configuration is a long text which is serialized and kept into a configuration listing, with the configuration assigned with a randomly generated identifier. Referring to figure 1, there are three main phases in the ADRO’s RL module and they are as follows;

1) Early learning phase: In the beginning, the agent has little or no knowledge about the DRE, so it needs to perform exploration by interacting with it through trial-and-error. For instance, where it has a MO module to search for the best configuration, the agent executes it to get the best configuration at the given time and workload. This module requires a substantial amount of system resources to execute.

2) Middle learning phase: After it has gathered enough knowledge about the DRE, it learns to predict the best action of configurations for a given DRE’s state to get the best reward. There is a high chance that it will guess incorrectly; where the predicted config_ID may not exist. In such cases, the agent’s RL module will invoke the MO module to do the configuration optimizing routine. It is expected that this will occur more frequently at the beginning of the middle phase. Its occurrence will start to diminish gradually as it gains more knowledge and its prediction is getting better when the results match with those in the configuration listing. The agent then adds into the knowledge base and the Q-table of \( Q(s,a,r,s') \), where Q value is a normalized value of the replication timing cost of r for the configuration used, a, against a DRE’s state, s, which will result in a new state, s’. Here we are concerned with the immediate reward as the DRE is highly dynamic and our objective is to meet the current needs. The knowledge base in which the agent stores the details also bolsters the minibatch which feeds into the NN predicting model as the training dataset. As it accumulates more data, the NN’s predicting accuracy will improve gradually.

3) High learning phase: By this stage, the agent would have learned all the states that exist in the DRE and able to predict the best configuration settings for the data replicating software that can yield the best reward with high accuracy. This is regarded as the exploitation of its vast build-up knowledge where the agent can provide a very quick turnaround time of activating the best configuration for any given DRE’s state to give optimum throughput. This process is outlined in Algorithm 1 and figure 1.

**A. Database and Shareplex Data Replication**

There are many types of databases that are currently available in the industry, and Oracle has been chosen to support the proposed method. It has several features that are important in supporting the functionality of this method and it also follows the IT system in which the testing dataset is drawn for the experiment later. There are several key DB features: 1) DB Replay is one of Oracle's Real Application Test options [22]. It allows the workload on the databases to be recorded and replayed either in situ or on other DBs [23]. 2) Flashback DB is a feature that allows the DB to reset back to a pre-defined point in time by rolling back the changes [24]. There are several types of Data Replication (DR) software available such as Oracle’s GoldenGate [4] or SAP’s Smart Data Integration [25] but Shareplex was selected to support the proposed method due to its history in supporting this operation in the space of IT enterprise’s near real-time replication and in line with the source IT system where the experimental dataset is obtained from [3]. It runs on top of the DBs, continuously captures all the data changes activities on those marked tables and propagates them to the target DBs in near real-time.
The Shareplex’s framework comprises several components that perform the functions of data capture, read, export, import, and post across multiple DBs as shown in figure 2 [26]. The main key focus for this proposed method is on the Shareplex’s configuration setting which dictates how many tables and queues are involved in the replication as illustrated in figure 3. The configuration setting contains a list of the replicating tables including their respective source and target details in the format of (target_system:named_queue@o.Target_oracle_sid) as shown in table 2 [27].

B. Meta-Heuristics Optimization for Shareplex

The Simulated Annealing (SA) is an analogous method for optimization that attempts to find the global optima among the large landscape of local optima of solutions for a problem environment [28]. Referring to Algorithm 1, annealing is a process where metals are heated and then cooled down to a hardened condition, with the objective function represents the energy of the material. The SA has a similarity with a hill-climbing algorithm [28] with the exception that it does not pick the better move in its iteration but rather a random one [5]. If the selected move can improve the solution, it will be accepted. However, this single goal can cause the algorithm to be stuck in local optima. Hence, the algorithm also takes a chance in choosing to accept a worse move based on some probabilities of a value that is less than 1. It starts with a high probability which means that the algorithm will be more liberal to accept the bad move but that will decrease rapidly with the degradation between the past and present moves [5].

Algorithm 1- SA method for finding optimum Shareplex’s configuration

```
#procedure to resync all tables at target site to source
sp_restore_source_tables
#procedure to restore all tables at source site
end

if (cost_new - cost_old) >=0 then
    cost_old = cost_new
end

#procedure to generate new configuration file
S = sp_generate_new_config

#test probability to accept new solution
A = 1 - e^(- (E new - E old) / T)
if A > random(1) then
    S = S'
end

T = T * decay_rate
```

The probability of accepting the uphill move is decided against random of (1) and it is represented as 1 - e^(- (E new - E old) / T), where E_new is the amount of energy at the present, E_old belongs to the previous iteration’s energy. T determines the probability and is synonymous with the annealing's temperature. It controls the algorithm's decision to take on the hill-climbing attributes for the moves, starting from a high-temperature value of T. The SA algorithm will be open to accepting the hill-climbing process. But as T decreases over time in energy, this probability will decrease, and the algorithm will be less inclined to accept this until T reaches zero or stopping criterion. k is the constant that relates the temperature to the energy [29]. The SA is commonly used to solve the optimization problem in large and discrete configuration spaces that can be considered as nondeterministic NP-complete problems such as traveling salesman problem, where the combination space of queues to tables arrangement is relatively large and an approximating global solution is required with a specific period of computational time [5].

Algorithm 2- Main Data replication's configuration optimizing Algorithm

```
Input: The state of the DBs
Output: optimum action's configuration and cost
Initialization1: set value for learning, reward preference, decay_rate, including rate and limit for exploration, learning, and exploitation
Initialization2: initialize memory, Knowledgebase, configuration list, and respective counters
Acquire the state from the DRE
Set the learning rate to zero, med_learning to 40%, high_learning to 90%
Loop the iteration process
    Check the learning rate.
    * exploration phase */
    If learning <= med_learning, do the exploration phase
    Run configuration optimizing search routine
    *reset the DRE environment. Apply Configuration on Shareplex. Run the workload
    Get the replication timing cost.
    Assign config_id to configuration & Store in configuration list
    Update knowledgebase (state, action, reward, new_state) /
    * learning phase & exploration phase */
    If learning > med_learning and < high_learning, then do
    Run configuration optimizing search routine */exploration phase*/
    Else
    Update the action's cost against DRE
    Update knowledgebase (state, action, reward, new_state)
    Else
    Run configuration optimizing search routine */exploration phase*/
    Update knowledgebase (state, action, reward, new_state)
    */ Knowledge exploitation phase */
    If learning > high_learning,
    */refer to the knowledgebase for action to state. /*
    If exploration_rate < exploration_limit then
    Exploits the knowledgebase to find optimum action's configuration for a given state to give the best rewards
    Else
    Run configuration optimizing search routine */ exploration phase*/
    learning_rate +=1
    exploration_rate= exploration_rate = decay_rate
```

C. Defining the State of the Environment

For the DRE, it is hard to define its state due to its complex multi-tier software setup and the characteristics of the IT applications under its service. A direct method is needed to identify a state in a database without time properties. Each database exhibits different characteristics throughout a period while they are in operation. Using the inspirations from Vizdoom and AlphaGo-Zero where the convolutional neural network is used to recognize the environment’s state [30], we propose the use of a matrix to capture a list of the following top attributes and events that occur in a DB during the specific period as a state’s representation. All the attributes are in numerical values with both the UserIDs and ObjectIDs that are easily obtainable.

The SQL statements are pre-processed with all the bind variables and spaces removed before passing them into a hash function: 1) Top list of users’ sessions that are the most active. 2) Top SQL statements that have been executed and incur the most writes. 3) Object IDs of the topmost accessible tables. To represent the state of DRE for a period, the DB’s matrices are combined with the hashed value of the pre-processed
Shareplex’s configuration setting as followed in the form of [DB1].[DB2].[Config] in eq(1);

\[
DRE's\ state,s_t = \begin{bmatrix}
j_{11t} & j_{12t} & \ldots & j_{1nt} \\
k_{11t} & k_{12t} & \ldots & k_{1nt} \\
j_{21t} & j_{22t} & \ldots & j_{2nt} \\
k_{21t} & k_{22t} & \ldots & k_{2nt} \\
j_{31t} & j_{32t} & \ldots & j_{3nt} \\
k_{31t} & k_{32t} & \ldots & k_{3nt} \\
j_{41t} & j_{42t} & \ldots & j_{4nt} \\
k_{41t} & k_{42t} & \ldots & k_{4nt}
\end{bmatrix} g_t \quad (1)
\]

The period of the state is t, user identification is j, the hashed SQL statement is k, identification of the object used is l, and the normalized value of the replication throughput is g.

D. Estimating the Cost of the Solution

The solution to the DRE’s environment comes in the form of a configuration setting that controls the number of queues the replication software creates to service the replication of the tables among the DBs. The configuration setting has a substantial impact on the overall replication’s efficiency and its throughput is the time required for the software to transfer the data changes from one DB to another. The goal is to find the best arrangement of queues and tables for the software to achieve the best data transfer in the least amount of time for a specific load of data changes. In the real-world scenario, the workload that creates the data changes’ loads is not constant and they fluctuated throughout the business operation periods. It is difficult to estimate the replication’s capability as there are other influencing environmental factors such as the following [3]: 1) the number of tables and the volume of data changes that are applied to them, 2) the number of queues that have been allocated, 3) the type of data that have been modified which range from numerical to large binary objects. But the strength of the configuration lies in its weakest link, which is the queues that take the longest time to clear the data transfer. Each of the replicating queues provides two sets of information that show its activity: The number of data or SQL statement changes, and the time remaining for the queue to clear them. In this proposed method, we sum up all the queues’ time spent to derive the overall result. So, the cost of configuration is the reward, r, for time, t, using queues, n, as is followed. The importance of each queue is associated with a weight, w. They are assigned to the value 1 as in Eq (2);

\[
r_t = \sum_{i=1}^{n} q_i w_i \quad (2)
\]

But for this test, we set a limit to narrow down the scope and improve the overall testing turnover. This factor will be available to be incorporated into the future project’s expansion or enhancement. The algorithm for finding the optimum configuration setting in a single cycle of RL is depicted in algorithm 2 as followed. The action’s solution is the configuration setting for the Shareplex and they are serialized for storage convenience. A configuration identifier that is comprised of a randomly generated number, is assigned to each one of them and they are stored in a configuration list. The NN associate the DRE’s states to the configuration IDs that can give the best replication throughput, which is regarded as a reward here.

IV. EMPIRICAL ANALYSIS

In this section, we discuss the test setup and the results. There are four groups of tests, and they are arranged in a hierarchical format starting with the benchmark test of the proposed system with best practices or stochastic methods that are commonly used in the domain of data replication. The next group test is to assess the difference in the approach of using adaptiveness versus static and non-adaptive approaches on the data replication using different metaheuristics. The third test is to study the time taken between the two phases of data replication throughput versus the computational time used that are encountered in each setup of the adaptive testing with different metaheuristics. The last test is more focused on the various outcomes against different workloads using just one optimizing algorithm.

The experiments were conducted with two virtual machines, each has 4GB of memory and 1 CPU, running on Linux OS with an oracle database and Shareplex instance. The Shareplex has been configured to replicate the tables under a schema from one DB to another. The tables have columns that comprised of different datatypes ranging from numeric to text. To simulate the workloads on the DBs, the alternative is to create SQL procedures that create the required SQL statements to do the data changes against the tables instead of using DB replay. The maximum number of queues specified for this test is 5.

The cost of a configuration setting in the experiments is the time taken for all the queues to export and apply successfully all the data changes from the source to the target DB. The measurement is not on the average time used by all the queues but rather on the queue that takes the longest time to clear.

Four different batches of workloads are used. The first set comprised INSERT statements which increase in the set of 100s for each table, totaling at 1 million entries. The second set has similar INSERT statements only but with random volumes assigned to each table, totaling to 1 million entries. The third set has a random set of equally divided INSERT, UPDATE and UPDATE statements and total to 1 million entries. The last set has a similar workload to batch #3 but the volume of the SQL statement increased by 50%.

There are two types of configuration setting tests; the first type is to show the nature of non-adaptiveness optimization. This is done by using getting the response time from the data replication software by using the same configuration setting for different batches of workloads. The second type of test is to use the agent with a different type of optimizing algorithm against the four batches of different workloads and gauge their response time. A total of 100 runs are conducted that are split equally among the 4 batches in runs of 1-25, 26-50, 51-75, and 76-100.

A. Benchmark test against other replication optimization practices

For this test, we use the various types of configuration optimization strategies that are commonly used in both academia and industry, ranging from using the out-of-the-box standard feature to methods of best practices recommendations from installation manuals [14], experienced IT administrator’s knowledge [31], whitepapers and best practices [15], including economic models [20, 32]. Their results are compared in line with a run’s results that use a configuration setting that has been properly optimized by the proposed method. Figure 4 shows the results. For the first run, we used the standard feature that is available under the common software setup [14] with all the replicating tables allocated to one single queue. Its results, labeled as “no-optimization”, showed that it has the highest values of replication throughput. This is because every table’s replication is serviced by one single queue and that has the longest delay. For the second run, the standard configuration setting is used with the just minimum use of two queues regarding best practices guides [15]. The tables are randomly
assigned tables to them without any consideration, with the ratio of the tables’ split is 30/70. Although the replication’s throughput has improved significantly by splitting the loads across to more than one queue, delays are significant as some tables with the large volume of data changes are grouped together and caused Shareplex to take a longer time to clear. The third run uses a similar technique [15] to the second run with the same number of queues with the number of tables evenly distributed across at the ratio of 50:50 without taking the volume of data changes’ load into consideration. The results showed more improvement over the previous runs.

B. Benchmark Test between Adaptiveness Versus Non-Adaptive Configurations with Different Heuristics

For the configuration optimizing routine, the control parameter for the simulated annealing algorithm is set with a high starting temperature of a value of 1000 and a smaller cooling rate of 0.9 to allow a better chance of finding the optimal solution. However, it requires more iteration and time to execute. The algorithm terminates when the temperature reaches a value of 1. The solutions are configuration settings generated via an algorithm and they are produced in each iteration under the SA’s temperature diminishing cycle. Each solution, or configuration, is applied to the source Shareplex in the DRE and followed by the execution of the workload scripts. The entire time used by Shareplex to completely replicate the data change from the source to the target will be the cost of the solution and it is normalized to the total minutes available in a day, i.e., 1440.

To justify the needs for adaptive optimization, we need to show that a configuration setting that has been optimized for one specific nature of workload is not suitable for the other types that have different load characteristics. For this test, the results are plotted in figure 5 by where the batch refers to the first test using the same configuration that has been optimized for batch #1 for all the runs. The other three are from the ADRO agent that uses three different types of optimizing algorithms, starting with Simulated Annealing (SA) and compared to other algorithms such as Hill Climbing (HC) and Tabu-Search (TS) [33].

A. Benchmark of Replication and Computation Test Using SA Against Other Heuristics

The next test is to assess the time taken for data transfer versus the computational time required for the configuration search. Three separate tests have been performed using a common SQL workload and the agent has been reconfigured to use each of the optimizing algorithms; SA, HC, and TS, in its optimum policy, with figure 6 showing the results of the test. The replication time in the figure refers to the time taken for each batch of workload to be completely replicated from the source to the target DB using the optimum configuration. The computing time refers to the duration for each optimizing routine that the agent has to perform to find the best configuration for a given workload, and that factor consumes a considerable amount of time and computer resources. All three charts showed similar RL behavior which is expected of different phases of interacting with the DRE.

The groups of results in figure 6 show similarity in terms of agent behavior and scores, despite using different optimizing algorithms. Figure 6 showed the consolidation of the computational time taken by the agent used using different algorithms and all follow the same patterns and cost, as each iteration requires generation and validation of 80 randomly generated solutions in the sub-iterations. The key difference is the replication timing where the scores from those that use the solutions’ configurations found by the HC algorithm have had higher values as compared to those searched by SA and TS as shown in figure 10.
algorithms used. So, the timing recorded among the different RL phases under the agent follow a common trend among the three batches of tests. For those iterations below the 100th range, the cost remains actively high due to the RL’s phase of performing solution search in every iteration to build up the knowledgebase.

The greedy algorithm which balances the chance between the actions of knowledge’s exploitation versus exploration in the agent does decay gradually, and there is a small chance that there may be a state that the agent has not encountered before. Therefore, that might invoke the exploration routine once again. In summary, the greatest benefit here is that the agent can save a lot in time and computing resources in managing the DRE when it starts to get more accurate in its prediction and better consolidation of its knowledgebase, especially when the DRE’s conditions and workloads start to change.

Referring to figure 7 and 8, for the replication throughput test, the results from the batch using Tabu-search is nearly the same with those obtained from the SA algorithm. Tabu-search also showed that it can meet the same requirement of finding the optimum solution within a limited sub-iteration underneath the main agent’s operating iterations. The results from using the Hill-Climbing algorithm show higher readings as compared to the other two. One of the main shortcomings with HC is that it tends to get stuck in local maxima and may not acquire global maxima whereas both SA and TS have the logic of avoiding getting stuck in local maxima and have a greater chance of finding the global maxima. The deviation between the results using different algorithms also attribute the insufficient complexity involved in the data replication nature which didn’t impose higher demands on the underlying replication services and therefore, less optimum configuration settings may not contribute to the overall performance. Future tests should include a higher number of tables with more variety of SQL workloads and data changes so that this can illuminate and distinguish the differences in the data replication’s setup with greater visibility.

Fig 8 – Comparison of replication time using configuration searched by different algorithms

Fig 9 – Generated random solutions’ costs under a single iteration

Fig 10 – SA’s selected solutions’ costs

B. Results from SA Heuristics Against Different Workload Setup

The next section discusses the result of solutions generated under the optimizing routine using the Simulated Annealing algorithm. Figure 9 showed the cost of all the randomly generated configurations in this routine and validated in the DRE for 5 cycles as a sample. Each cycle has 80 iterations with an identical workload that comprises a series of SQL INSERT statements that span across 10 tables and their volumes increment hierarchically. The optimizing routine follows the SA algorithm and selects the configuration that gives the best cost from the secondary iterations that occur under each primary iteration. Such results are collected and shown in figure 10. The result may not be optimal in the 1st cycle but the result starts to converge on the 60th iteration in the 1st cycle onward, with the subsequent 2nd to the 6th cycle showing full convergence. Figure 10 is a subset of an iteration in Figure 11. Figure 11 shows the application of the optimized configuration setting for the Shareplex data replication for 5 cycles of workloads, each with similar characteristics of SQL INSERT statements but varying in volumes across the 10 tables - totaling to 1 million rows of INSERTs. The result from all the cycles may not be similar, and this small difference observed can be attributed to the system set up in which the DRE runs off from - a guest Virtual Machine that is dependent on its host which also has its hardware and background process influencing factors.

C. Comparative Performance of ADRO with other methods

There are several methods of setting the configuration for the data replication environment as listed in Table 1, ranging from the standard out-of-the-box default setup that comes with the installation, to rule-based decision processes. All of them do not consider the business of data replication nor the characteristics of the data replication landscape. The latter goal helps to increase the throughput by splitting the workload across to multiple queues and this is done with a general assessment to gauge the performance. The next set of the replication’s optimization comes in the form of employing statistical methods such as economic models that try to factor in various parameters or variables from the replication’s setup and tables’ attributes to find an equilibrium between the resources of demands versus supply among the replications’ workloads versus the throughput threshold. We compared our approach’s results against the current practices of optimizing the data replication systems that are listed in table 1. Based on the Delphi method, we present our method as well as the others to a group of experienced IT administrators and gather their feedback. The results are ranked against several criteria in table 3 below.

V. Conclusion

We present a novel machine learning-based approach, ADRO, that can adaptively and autonomously optimize a data replicating software across two databases over a period with constantly changing IT workloads. ADRO alters the
configuration setting of the Shareplex setup constantly to keep up with the evolving workload and user activities to keep it operating at the best performance. The optimizing routine that is performed by ADRO is a resource-intensive and time-consuming process that ideally should not be repeated too frequently. Therefore, ADRO learns the situation in the DRE and adapts to alternate the configuration setting when the workload changes based on its knowledgebase without mindlessly re-initiating the tuning search routine every time it needs to alter. Empirical analysis shows that it can adjust the configuration in response to the changing data replication workload. Consequently, it is handling the changes on the DRE in an intelligent manner with minimum human interaction.

### References
1. King, E., *Automated Database Refresh in Very Large and Highly Replicated Environments*. 2011.
2. Simitsis, A., P. Vassiliadis, and T. Sells. *Optimizing ETL processes in data warehouses*. in *null*. 2005. IEEE.
3. Software, Q., *Shareplex 9.0 Reference Guide*. 2018.
4. Gupta, R., *Introduction to Oracle GoldenGate (OGG)*, in *Mastering Oracle GoldenGate*. 2016. Springer. p. 3-10.
5. Chopard, B. and M. Tomassini, *An Introduction to Metaheuristics for Optimization*. 2018. Springer. p. 59-79.
6. Software, Q., *SharePlex 9.0 - Reference Guide*. Quest Support, 2018.
7. Soumyav, S. and A. Sifaleras, *Trends in data replication strategies: A survey*. International Journal of Parallel, Emergent and Distributed Systems, 2017: p. 1-18.
8. Hamdani, C., T. Hamrouni, and F.B. Charrada, *Evaluation of site availability exploitation towards performance optimization in data grids*. Cluster Computing, 2018. 21(4): p. 1967-1980.
9. Nazir, B., et al., *The Impact of the Implementation Cost of Replication in Data Grid Job Scheduling*. Mathematical and Computational Applications, 2018. 23(2): p. 28.
10. Blum, C. and A. Rolli, *Metaheuristics in combinatorial optimization: Overview and conceptual comparison*. ACM computing surveys (CSUR), 2003. 35(3): p. 268-308.
11. Goss, R.G. and K. Veeramuthu. *Heading towards big data building a better data warehouse for more data, more speed, and more users*. in *ASMC 2013 SEMI Advanced Semiconductor Manufacturing Conference*. 2013. IEEE.
12. Russo, D. and B. Van Roy, *Learning to optimize via posterior sampling*. Mathematics of Operations Research, 2014. 39(4): p. 1221-1243.
13. Habibi, A., A. Saraffrzi, and S. Izadyar, *Delphi technique theoretical framework in qualitative research*. The International Journal of Engineering and Science, 2014. 3(4): p. 8-13.
14. Software, Q., *Shareplex 9.0.1 Installation Guide*. 2019.
15. software, Q., *SharePlex 9.0 - Administration Guide*. 2019.
16. Wee, C.K. and R. Nayak, *Data replication optimization using simulated annealing*. in *Australasian Conference on Data Mining*. 2019. Springer.
17. Van Hasselt, H., A. Guez, and D. Silver, *Deep reinforcement learning with double q-learning*. in *Thirtieth AAAI conference on artificial intelligence*. 2016.
18. Wee, C.K. and R. Nayak, *Adaptive load forecasting using reinforcement learning with database technology*. Journal of Information and Telecommunication, 2019: p. 1-19.
19. Kenong Wee, C. and R. Nayak, *A novel machine learning approach for database exploitation detection and privilege control*. Journal of Information and Telecommunication, 2019: p. 1-18.
20. Carman, M., et al. *Towards an economy-based optimisation of file access and replication on a data grid*. in *2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID’02)*. 2002. IEEE.
21. Sivaganesh, K., et al., *Optimization of ETL Work Flow in Data Warehouse*. 2012. 4(9): p. 1579.
22. Colle, R., et al., *Oracle database replay*. Proceedings of the VLDB Endowment, 2009. 2(2): p. 1542-1545.
23. Ngai, G., et al., *Automatic workload repository battery of performance statistics*. 2009. Google Patents.
24. Kuhn, D., S. Alapati, and A. Nanda, *Performing Flashback Recovery*, in *RMAN Recipes for Oracle Database 12c*. 2013. Springer. p. 395-442.

### Table 3– Benchmarking RL Tuning with Other Methods

| Methods | Complex | Skill | Effort | Adaptive to changes | Repetitive manual effort | Tuning coverage | Replicability | Optimum result |
|---------|---------|-------|--------|---------------------|------------------------|----------------|-------------|---------------|
| Out-of-box setup configuration | V low | Low | None | High | None | High | None | Very slow | None |
| Manual [15] | Low | Low | None | High | None | Low | Med | Low |
| Best practice, Decision Rule-based [15] | Low-med | Low-med | None | High | None | Low-med | Med | Low |
| Economic Model-based [20] | Med-high | Med-high | Low | High | Med | Med | Med | Med |
| Metaheuristics Optimization [21] | High | Low | Med | Med-high | Low-med | Med | Med | Med |
| Adaptive RL optimization tuning | V High | Low-med | High | low | High | Fast | Fast | None |