From: Computing Conference
Sent: Saturday, December 18, 2021 4:33 AM
To: Tammy Fuller <tammy@echomessaging.com>; Gerald Deane <gerald@echomessaging.com>
Subject: Acceptance Notification : Computing Conference 2022

Dear Tammy R Fuller, Gerald E. Deane,

Congratulations, your submitted paper "Quantum Computing in Agent-Based Technologies" has been accepted for inclusion in the Computing Conference 2022, to be held from 14-15 July 2022 in London, UK

We are of course aware that the situation regarding COVID-19 is a cause for apprehension - virtual participation (with reduced registration) is available, for anyone who cannot or chooses not to travel.

Computing Conference is a research conference held in London, UK since 2013. The conference series has featured keynote talks, special sessions, poster presentations, tutorials, workshops, and contributed papers each year. The goal of the conference is to be a premier venue for researchers and industry practitioners to share new ideas, research results and development experiences in various fields.

Computing Conference 2022 proceedings will be published in Springer series "Lecture Notes in Networks and Systems" and submitted for consideration to Scopus, Web of Science, DBLP, INSPEC, WTI Frankfurt eG, zbMATH, SCImago.

The Conference Board has decided that the reviewers' feedback and invitation letter for visa applications (if necessary) will be emailed to the author(s) after the registration process. If you would like to receive the reviewer feedback for representation at your university/organisation, please feel free to contact us.

Again, congratulations and I look forward to your participation!

Regards,
Kohei Arai
Program Chair
Computing Conference 2022

View 2020 Recap

Accepted to IEEE sponsored Computing Conference 2022, 14-15 July 2022 – London, United Kingdom

Quantum Computing in Agent-Based Technologies

Tammy R. Fuller[1] and Gerald E. Deane[2]

[1] Chief Architect/VP, Echo Messaging Systems, Inc. 198 Chapel Street, Lincoln, Rhode Island 02865, USA
tammy@echomessaging.com
[2] CEO/President, Echo Messaging Systems, Inc. 198 Chapel Street, Lincoln, Rhode Island 02865, USA
gerald@echomessaging.com

Abstract. Agent-based solutions reduce complex applications into manageable components that run independently of each other. Quantum computing solves particularly difficult problems, such as Constraint Satisfaction Problems, that are time consuming on classical computers. Segmenting a complex problem into quantum and non-quantum components is well-suited to agent-based processing since the framework is already in place to simplify and solve through a divide-and-conquer approach. In this paper, we show how Quantum Agents in an agent-based intelligent system can be an ideal interface to quantum computing for solving many common and difficult to solve real-world problems. Quantum computing is being used for today's applications for optimization, resource allocation, scheduling and route calculations. Pairing quantum computing's advantages with the flexibility, integrity, and reusability of an agent-based system, produces application solutions quickly, that are reliable and evolve to meet expanding needs and growing scale.

Keywords: quantum computing, quantum, agent-based processing, agents, quantum agents, anomaly detection, intelligent notification, adaptive agents, intelligent system, constraint satisfaction problem.

1 ADIN – An Agent-Based Technology

1.1 Introduction

Agent-based programming has a long history of turning large complex problems into smaller manageable components by dividing up the tasks into a series of triggering events and associated action responses [1]. Each agent runs independently and is focused on its own triggering criteria, and once the criteria is met, will execute its associated action response. Agents react to triggers based on data, time, space (geo-location) or any combination. Once triggered, actions performed include creating/updating records in a database, issuing notifications, controlling processing and/or devices and much more [2].

Quantum computers, based on quantum gates are different from classical computers which are based on binary gates, in that a quantum gate can hold both a 0 and 1 as a value at the same time, up until the time a solution for all quantum gates is found. Quantum computers, such as D-Wave [3] solve problems through a process called 'quantum annealing' which generally finds the best solution over large multi-dimensional search

spaces. Problems are modeled as energy cost functions and constraints, as linear and quadratic equations are called Constrained Quadratic Models (CQMs). The CQM is sent to the quantum computer, which embed the formulated problem onto its topology of quantum gates. The quantum annealing process is performed on the embedded equation and it returns the terms to the quadratic equation that minimize the overall cost function while maintaining the constraints. This technique is well suited for constraint satisfaction problems that model as linear or quadratic equations and may have many constraints, such as job scheduling, resource sharing and allocation, optimization of systems of parameters, and route calculations [4].

Quantum computers are ideal for an agent-based system, where the complex problem is divided into components that are well-served by quantum techniques and the remaining to be solve via classical computer processing. This is known as a hybrid approach and by merging this concept with agent-based technologies, a hybrid approach is further advanced by the dynamic adaptive nature of agents.

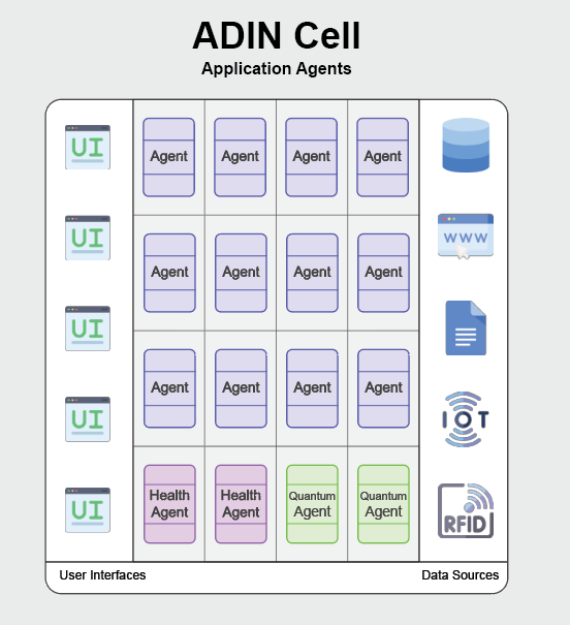


Fig. 1. An ADIN Cell is a grouping of ADIN Agents which can be designated for processing, monitoring health, interfacing with quantum processing, and more. Common user interfaces and data sources can be shared among agents.

In the following pages, we first give a brief overview of our adaptive agent-based intelligent system called ADIN as shown in Fig. 1. Next, we show how ADIN agent types called ‘Quantum Agents’ apply quantum processing techniques are used to optimize how all the agents in a cell are configured. Triggering criteria and action response components of agents both include quantum computing functions to meet application needs and we provide information and examples, including how push notification systems and geo-fencing can benefit from quantum computing. Finally, we explore how quantum computers, currently limited by variable sizes, which grow quickly based on the number of

constraints in the CQM, can benefit from agent-based optimization to decompose a CQMs automatically to fit these variable limits.

1.2 ADIN - Anomaly Detection and Intelligent Notification

ADIN stands for Anomaly Detection and Intelligent Notification [5] and is used to solve many problems and applications with common, but difficult to solve attributes that involve asynchronous or parallel processes. Agents are grouped into an ADIN Cell and share data sources such as databases, API's, IoT devices, and more. User Interfaces connect users to the ADIN Cell by providing points of input and showing results. Application supported by ADIN Cells range from communication systems, automated workflows, data synchronization and more.

Agents have parameters which can be adjusted by other agents, or themselves in response to the processing environment, making them adaptive. Agents can control other agents through cloning functions, which is helpful to expand or control resources in response to current data levels.

Agents are configured by defining their triggering criteria and action responses. Triggering criteria is based on Time, Space, Data, or any combination of these three categories. ADIN's Rest API means that agents are fully configuration from the type of data sources they connect to the type of actions that result. After agents have been instantiated, they can be stopped, and parameters adjusted and restarted.

For example, if the events being processed by the triggering criteria hit a throughput limit and aren't processing all the events in time, the agent can be paused, cloned and parameters the original agent set to process the first 50% of the data source and the newly cloned agent set the second 50%. Both agents are started and run independently of and parallel to each other. The throughput rate of the first (original) agent is cut in half and spread across two agents.

1.3 Types of Agents

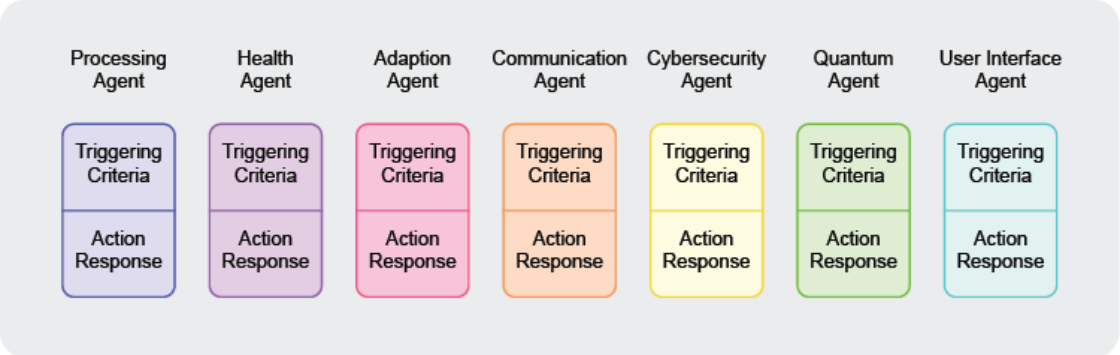


Fig. 2. Agents fall into categories based on how they are used in the ADIN cell. All agents are based on their triggering criteria and action responses.

Processing Agents do the main work of the application. As shown in Figures 2 and 3, ADIN has many types of agents, and many options for agents' triggering criteria and action responses.

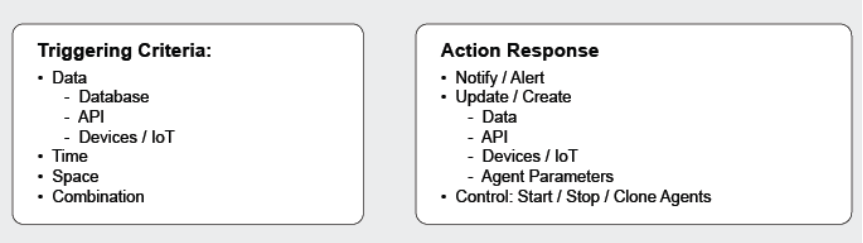


Fig. 3. Agents have 2 components: Triggering Criteria and Action Responses.

For example, a health and wellness application that uses the ADIN platform, must process 600 different types of trackable attributes over the course of the day for each user. Some attributes are based on sensor user data from wearable devices, where other attributes are provided directly from users for information about their daily health from sleeping, eating, stress events and daily circadian rhythm events. Some trackable attributes are timed events that occur because of other combination of events. Each agent is assigned a trackable attribute. The triggering criteria is configured to monitor the data source where the trackable attributed can be found. For 600 attributes, you have 600 agents.

The following Table 1 shows examples built using ADIN to meet a variety of application requirements.

Table 1. Applications using ADIN Cells.

Application	Agents used and examples
Automated Notification Systems	Agents monitor job statuses and notify customers, and technicians via email, text, and text. *Example: Echo Messaging Systems JNS – Job Notification System*
Automated Workflow Systems	Agents monitor event status and create new and/or update existing according to workflow rules and business logic. *Examples: Safe Zones – monitors & verifies requirements to enter geofences areas. Mira – acts as a daily timekeeper of your aging, health, well-being, and performance.*
Data Syncs	Agents monitor new and update records from one system and automatically replicates into another system. Data syncs are highly configurable and work in both directions.

	Examples: Infusionsoft – QuickBooks Desktop Data Sync. Sage 100 Pro/Premium – GoldTech POS Data Sync. Zoho – Magento2 – QuickBooks Desktop Bridge Module.
Push Notification with Geofencing	Agents monitor activity going into and out of geo-fenced regions. Action responses push data to mobile devices as they cross into geofenced region. *Example: Echo's Push Channel System*

ADIN agents are containerized [6]. This means that an agent runs inside of a Docker container and is orchestrated by systems like Kubernetes, which make sure there is enough computing power resources for all the agents. If the containerize agents require more and more computing resources to process the trackable attributes for a growing user base, then Kubernetes handles expanding the resources as agent requirements grow. Agents focus on their task of detecting their particular triggering criteria and when found, performs the action responses on the data the criteria returned.

Health agents have triggering criteria to monitor processing agents and trigger when the agent stops running. Health agents' action responses can be one of several options including restarting the agent, but also include notifying the ADIN cell administrator that an anomaly has occurred and needs attention.

Adaptation agents have triggering criterial that monitor how much data is being processed and if the parameter settings need adjusting. Communication agents monitor messages issued from agents and when they occur, transmit messages to the other agents. Cybersecurity agents have triggering criteria that monitor for anomalies that unauthorized processes are occurring. Containerized agents have more control over their environment that non-containerized and this reduces unauthorized activity. User Interface agents trigger off user input and action responses return user interface components html pages geared toward the user input request.

Quantum agents have parameters to support triggering criteria and/or action responses that involve constraint satisfaction problems. The next section will describe in depth how Quantum Agents can be used to solve many different types of applications.

2 Quantum Agents for Optimization

Optimization is one of most important ways quantum computers excel over classical computers [7]. By presenting a complex system of parameter settings, such as we find in ADIN Cells, we want to use quantum computing to find the base set of parameter settings to maximum the efficiency of agents to reduce wasted resources.

If we think of all the ways agent parameters can be set, such as, how many agents per agent event type vs. how many events it processes over time vs. how often agents are run, there will be a combination of parameter settings that minimize wasted computing resources. Some parameter settings will produce good results like we see in the 'local

minimum' in Fig 4 below. Using the quantum computing technique of quantum annealing, we can find the global minimum (i.e., the best way to setup agents). Because agents can control other agents through their action responses, we can configure a Quantum Agent to determine the optimized agents for a cell, then pause agents, clone agents, set agent parameters and start agents according to the optimized setup.

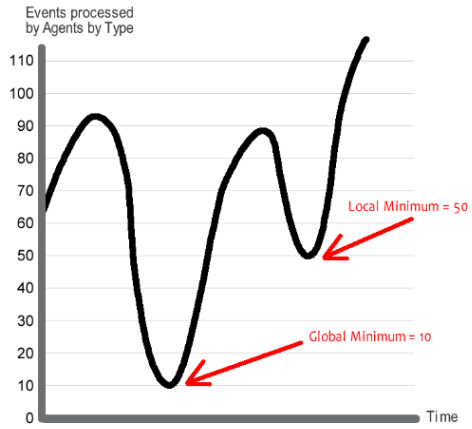


Fig. 4. Quantum annealing to find global minimum for a large, multi-dimensional search space

Quantum computers like those from D-Wave will solve global minimum problems through circuitry that performs quantum annealing and are used in many applications including green technologies [8], financial investment portfolios and drug discovery, among many others. For ADIN to make use of this type of optimization, we must first formulate the problem as a linear or quadratic equation expressed in terms of binary variables.

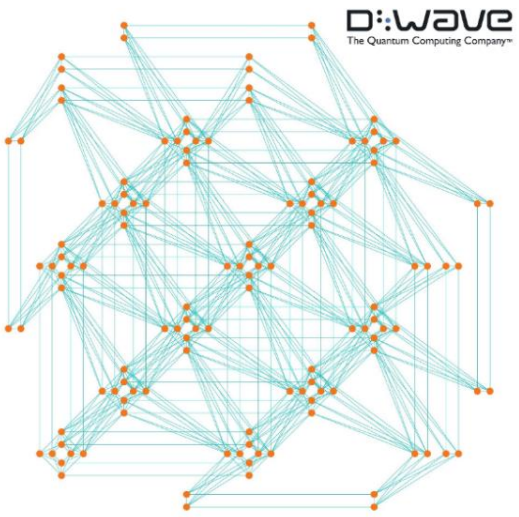


Fig. 5. DWave is a quantum computer. This diagram shows the topology of one of its quantum computers.

The equation model is embedded onto the quantum computer's topology of qubits and solved by quantum annealing. Fig. 5 shows one of D-Wave's topologies of qubits. It helps to understand at the qubit level how the variables can hold both a 0 and 1 and during the annealing process, eventually commit to a 0 or a 1 as its part in minimizing overall energy.

First, we total all the events handled by each event type, sum the processing time used by each event, and call this the load per event. Next, we setup the Constraint Satisfaction Model by creating a binary variable for every agent at every possible agent event configuration. If the binary variable is 1, that agent is set to that agent type. If it's 0, the agent is not set to that agent type. Every agent type must be represented at least one time.
A simple example is given 8 total agents and 4 agent event types, without any consideration of how many events of each type are triggering or how often each agent runs, you would assign 2 agents per agent event type where each processed 50% of the data. If the loads for the 4 types were (200, 50, 125, 25) for agent event types (A, B, C, D), A will get more than C. A small example can be determined thru a quick program that A will get 4 agents, B will get 1, C will get 3 and D will get but will be under resourced, to compensate for C being over-resourced.

```
num_agent_types = len(agent_types)
total_load = sum(loads)
LPA = int(total_load / max_agents) ## a lower priority constraint than the other 2 constraints

cqm = ConstrainedQuadraticModel() ## create Constrained Quadratic Model
bqm = BinaryQuadraticModel('BINARY')

# Build a variable for each agent at each possible agent type
bin_variables = [[Binary(f'A{str(i)}_{str(j)}')for j in range(max_agents)]
                 for i in range(num_agent_types)]

## add following constraints to Constrained Quadratic Model
for j in range(max_agents):
    ## MUST: each agent has one agent_type
    cqm.add_constraint(quicksum(bin_variables[i][j] for i in range(num_agent_types)) == 1,
                       label='c1_one_per_agent_'+str(j))

for i in range(num_agent_types):
    ## MUST: each agent type is represented in at least 1 agent
    cqm.add_constraint(quicksum(bin_variables[i][j] for j in range(max_agents)) >= 1,
                       label='c2_agent_type_'+str(i))

    ## BEST: sum of loads processed by agents of same agent type are >= to load reqs
    # for agent type
    c1 = [(f'A{str(i)}_{str(j)}', LPA) for j in range(max_agents)]
    bqm.add_linear_inequality_constraint(c1, lagrange_multiplier=1, label='load',
                                         lb=int(loads[i]), ub=total_load)
cqm.set_objective(bqm)
sampler = LeapHybridCQMSampler()
sampleset = sampler.sample_cqm(cqm, label='Optimize ADIN')
```

Fig. 6. Python code snippet creating a Constrained Quadratic Model, adding binary variables and constraints, and solving for the optimized configuration of agents.

The python code snippet shown in Fig. 6, show how to setup the D-Wave Constrained Quadratic Model, by setting up the binary variables, adding the row-based constraints, the column-based constraints, and the global constraints, then sending to the LeapHybridCQMSolver, which returns the values of the CQM terms for which agents are assigned to which agent types.

2.1 Mira Application

Mira is a health and wellness application built using the ADIN platform. Mira is an AI system that serves as a "Digital You", acting as a daily timekeeper of your aging, health, well-being, and performance. Your "Digital You" serves as an NFT that continually matures-- a digital asset that increases in value hour-by-hour every day.

What is the Digital You?

The 'Digital You' generates troves of actionable data by way of your circadian clock, 600 timed circadian events, a simulation of digestive system cycles, sleep-wake states, well-being, vital signs, health meters, health tracking device data, and algorithms that quantify the impact of disruptions, such as your daily stress events, against them; exposing some of the causes of premature aging and dis-ease.

Your Digital You is run by hundreds of AI ADIN Health and NFT Agents, each mapping and calculating changes that increase the value of your "Digital You NFT" hour-by-hour every day.

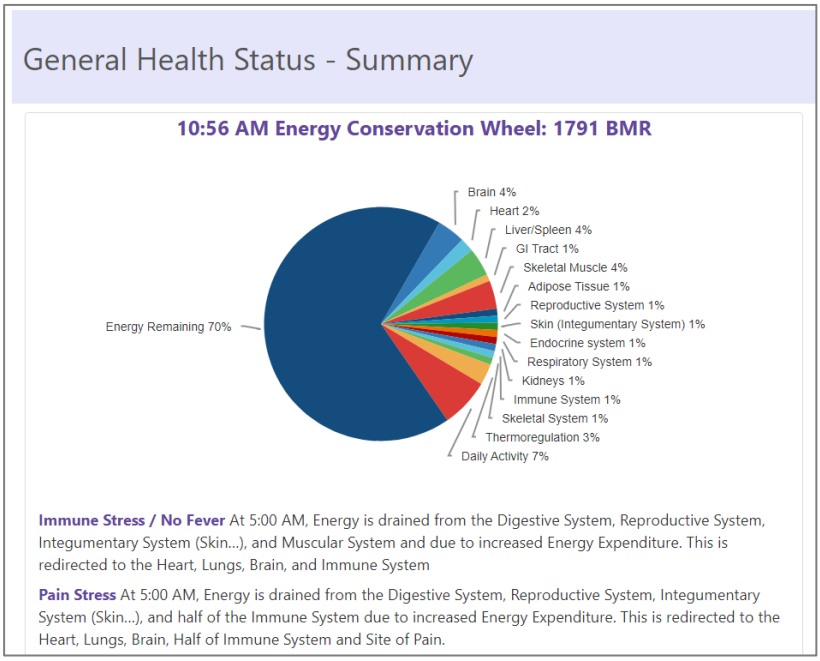


Fig. 7. Mira's Energy Conservation Wheel showing the impact of stress events as it relates to circadian events, and it sum effect on health, wellness, and aging.

This application is a great example of how using both agent-based processing and quantum computing will result in high functioning applications that scale well and retain their integrity as the application evolves. Mira is a Demo application currently processing a handful of event types that mirror stress events that can occur anytime during the day. This number will grow to 600 as the full application comes online, and even though this project is in the demo phase, all agent components are 100% online and 100% functional. The demo aspect is that Mira is tracking and mapping the energy impact of a subset of stress events as it relates to circadian events and shows through an Energy Conservation Wheel (Fig. 7) the sum of the impact stress is having on health, wellness, and aging.

3 Quantum Agents for ADIN Application Agents

3.1 Agent Triggering Criteria

Triggering criteria determines if agent will perform its associated action response and what data it will process over. ADIN current has agents that process over data, time, space (geo-location) and any combination. For example, agents trigger when a new customer is detected, when a job status changes from pending to active, when an IoT API returns an error warning, when a process has not run in the past 5 minutes, when a geo-tracked mobile devices crosses into a geo-fenced area. Triggering criteria can become much more complicated, such as between 8am and noon, any geo-tracked devices that cross into a geofenced area and have a pending job with an error warning can also be the triggering criteria. This phase returns the set of records to be associated action response.

Adding quantum processing to the triggering criteria, we can add triggers such as, trigger if the optimized parameter settings for an agent have changed. Another is to trigger if the resource allocation of items that have crossed into a geo-fenced area are no longer sufficiently distributed due to more devices entering the geo-fenced region.

3.2 Agent Action Responses

Agent's action responses that are expanded through quantum computing functions can be triggered by quantum or non-quantum triggering criteria. For example, an agent can trigger if a new job has been detected that has not yet been scheduled. The action response is to use quantum computing to best schedule the new job taking into account constraints based on currently scheduled jobs, job requirements and job deadlines. Another example is where an agent's triggering criteria says between 8pm and midnight, if a geo-tracked device crossed into a geo-fenced area, and the associated phone number of the device matches a completed job, will return the customer and their completed job. The associated action response is to use quantum computing to calculate a delivery route to bring the completed job to the current customer location.

3.3 Quantum Agents and Push Notification

Using quantum agents with push notification, a system were the geo-fenced regions position and size are changed to fit the people inside the region. A Constrained Quadratic

Model will indicate the goal of how the geo-fences areas are positioned and size as show in Fig 8.

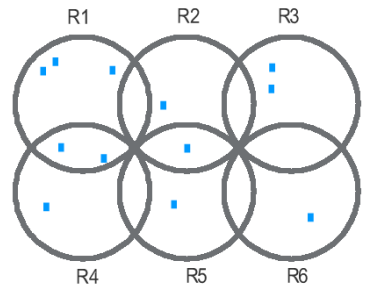
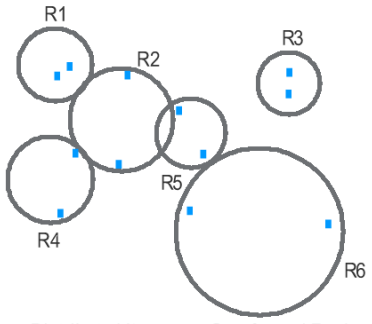
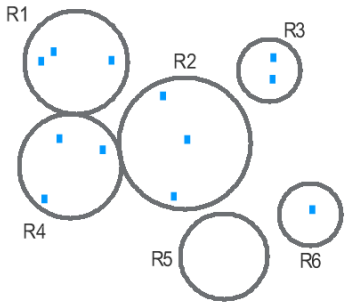
 Initial Geo-fenced Region Locations	This is an example of starting positions of geo-fenced region with no data or goals for positioning. Each geo-fenced region is assigned an ADIN agent that monitors activity in its borders and pushes data based on the application.
 Distributed Items per Geo-fenced Region	Once sensing information about people in each region is gathered, quantum agents can update the location and size of each agent to better meet the application 's goals. In this scenario, a quantum agent will reposition and resize geo-fenced regions to evenly distribute people for scenarios where pushing notifications to people within a region at the same time will maximize communication bandwidth per agent.
 Cluster Items into Geo-fenced Regions	Quantum Agents determine geo-fence regions based on clustering. If dots are people, dynamic clustering in real-time can predict areas that are too densely populated to predict surges of people.

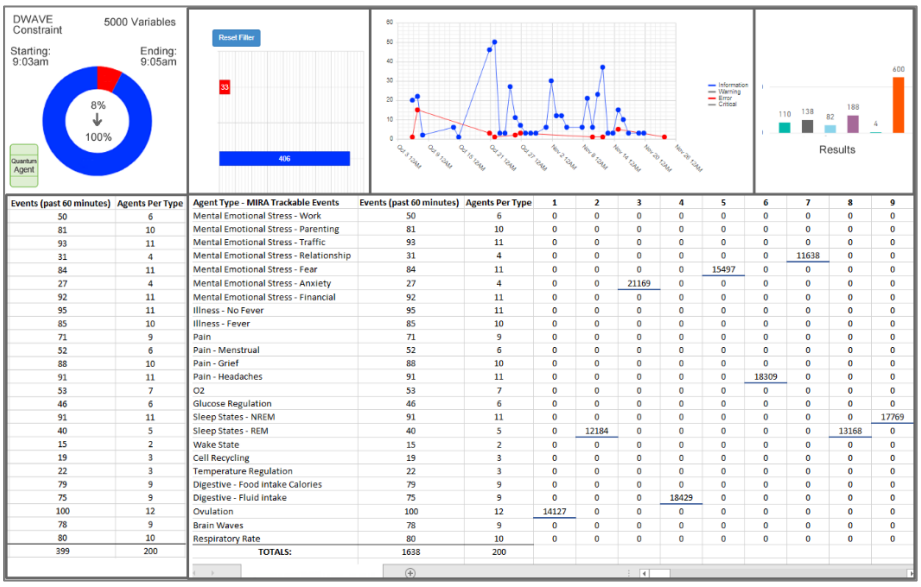
Fig. 8. Use CQM to position and size geo-fenced regions based on several different goals.

A dynamic system where geo-fenced areas that mapped to more people in smaller and smaller spaces could predict overcrowding conditions before they became dangerous and through push notification serve as an early warning system to prevent dangerous overcrowding from occurring.

3.4 Quantum Agents User Interfaces

AI often suffers from black-box related situations, where complex interactions are occurring but without insight as to how and why. Since agents breakdown problems, it's

possible to log and record statistics as they run, then provide user interfaces to show all aspects of agents.



Events (past 60 minutes)	Agents Per Type
50	6
81	10
93	11
31	4
84	11
27	4
92	11
95	11
85	10
71	9
52	6
88	10
91	11
53	7
46	6
91	11
40	5
15	2
19	3
22	3
79	9
75	9
100	12
78	9
80	10
399	200

Agent Type - MIRA Trackable Events	Events (past 60 minutes)	Agents Per Type	1	2	3	4	5	6	7	8	9
Mental Emotional Stress - Work	50	6	0	0	0	0	0	0	0	0	0
Mental Emotional Stress - Parenting	81	10	0	0	0	0	0	0	0	0	0
Mental Emotional Stress - Traffic	93	11	0	0	0	0	0	0	0	0	0
Mental Emotional Stress - Relationship	31	4	0	0	0	0	0	0	11638	0	0
Mental Emotional Stress - Fear	84	11	0	0	0	0	15497	0	0	0	0
Mental Emotional Stress - Anxiety	27	4	0	0	21169	0	0	0	0	0	0
Mental Emotional Stress - Financial	92	11	0	0	0	0	0	0	0	0	0
Illness - No Fever	95	11	0	0	0	0	0	0	0	0	0
Illness - Fever	85	10	0	0	0	0	0	0	0	0	0
Pain	71	9	0	0	0	0	0	0	0	0	0
Pain - Menstrual	52	6	0	0	0	0	0	0	0	0	0
Pain - Grief	88	10	0	0	0	0	0	0	0	0	0
Pain - Headaches	91	11	0	0	0	0	0	18309	0	0	0
O2	53	7	0	0	0	0	0	0	0	0	0
Glucose Regulation	46	6	0	0	0	0	0	0	0	0	0
Sleep States - NREM	91	11	0	0	0	0	0	0	0	0	17769
Sleep States - REM	40	5	0	12184	0	0	0	0	0	13168	0
Wake State	15	2	0	0	0	0	0	0	0	0	0
Cell Recycling	19	3	0	0	0	0	0	0	0	0	0
Temperature Regulation	22	3	0	0	0	0	0	0	0	0	0
Digestive - Food intake Calories	79	9	0	0	0	0	0	0	0	0	0
Digestive - Fluid intake	75	9	0	0	0	18429	0	0	0	0	0
Ovulation	100	12	14127	0	0	0	0	0	0	0	0
Brain Waves	78	9	0	0	0	0	0	0	0	0	0
Respiratory Rate	80	10	0	0	0	0	0	0	0	0	0
TOTALS:	1638	200									

Fig. 9. Quantum Agents dashboard

This is being expanded to include all interactions with quantum computing functions as well. Fig. 9 shows the Quantum Agent user interface areas linked data can be further inspected as shown in Fig. 10.

Fig. 10. Quantum Agent Details dashboard

3.5 Agent-based Quantum Model Decomposition and Reconstruction

Quantum computers map large multi-dimensional problems by embedding linear and quadratic terms onto qubits to hold both 0's and 1's until the equation's overall energy is minimized giving its optimal solution. Problem formulation tells which parameter dimensions are needed as well as constraints on the parameters. As more constraints are added, the number of variables needed to be mapped on to the quantum computer are also increased. Quantum computers are limited to several thousand qubits. Techniques to improve solutions [9] have been discussed for some but not all solvers, and product roadmaps such as D-Wave quantum computer will be increasing in number of qubits and just Moore Law, which observed with classical computers circuit density doubled every two years, quantum computer will increase as well.

Our health and wellness application that has 600 different agent event types running as containerized agents will require thousands of agents running simultaneously to service a growing user base. If we assume 10,000 agents running 600 different agent event types, where we want to optimize how many, and how often they run, where each agent type is represented by at least one agent, our CQM contains 6 million variables. If the quantum computer has a variable limit of 5000, we need to decompose the entire problem into 1200 units.

To accomplish this, we propose defining the original Constrained Quadratic Model as an ADIN Quantum Matrix, as shown in Fig. 11. A Quantum Agent will solve for clustering the parameter dimensions to decompose the ADIN Quantum Matrix into ADIN Quantum Cubes (step 1).

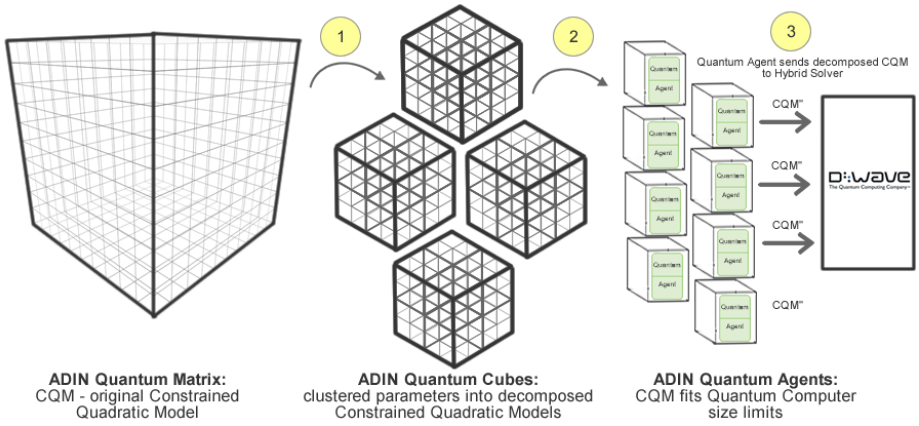


Fig. 11. ADIN Quantum Matrix Decomposed into CQM's that fit quantum computer

Every ADIN Quantum Cube instantiates another Quantum Agent that likewise clusters the multidimensional parameter space into smaller and smaller ADIN Quantum Cubes (step 2) until its number of variables fits the quantum computer's size limits.

Once the CQM for a Quantum Agent fits the quantum computer's size limits, the agent submits the CQM to D-Wave's hybrid solver.

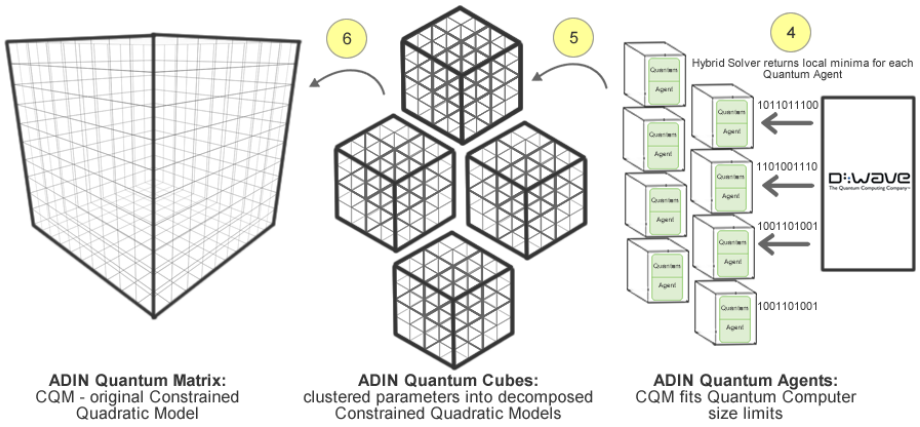


Fig. 12. ADIN Quantum Matrix Reconstructed from local solutions returned by ADIN Quantum Cubes into a global solution for original CQM

The hybrid solver returns the CQM for each Quantum Agent as shown in Fig. 12, that represents the solution for that portion of the total problem space. Agents instantiated during the composition process are created to monitor for results returned from the hybrid solver (step 4) as they appear. Quantum Agent results are propagated back to each's associated Quantum Cube (step 5) where likewise agents reconstruct by merging local minima back to the original ADIN Quantum Matrix to return the final global solution to the original CQM (step 6).

This algorithm of applying quantum agents to decomposing multidimensional parameter space into clusters recursively is known as the "Deane Fuller Quantum Matrix Algorithm".

4 Conclusion

Quantum computers reduce the complexity of difficult problems involving Constraint Satisfaction by formulating the problem as a quadratic model. Agent-based adaptive systems reduce overall complexity by dividing the problem into small manageable components based on triggering criteria and action responses. Merging these two technologies together to create Quantum Agents, means applications benefit from quantum computing resources while letting the agents do the direct interfaces to the quantum computer. ADIN agents as an interface to quantum computing will reduce the steep learning curve associated with quantum computing, by focusing on particular use cases involving job scheduling, resource allocation and sharing, logistics routing and optimization.

4.1 Quantum Computers in an Agent-Based Intelligent System

As quantum computers' qubit numbers increase, Constraint Satisfaction Problems with more and more parameter dimensions can also increase. To solve problems larger than what quantum computer can handle due to variable limits on one single Constrained Quadratic Model, we propose creating an ADIN Quantum Matrix. A Quantum agent is used to solve a multi-dimensional parameter clustering which determines how to break the

problem down into Quantum Cubes using automated agent-based processing to decompose into small enough CQM's. Once small enough, the CQM is submitted to the quantum computer. Results returned from the quantum computer are automatically reconstructed to return the global solution to the original CQM.

Agent-based technologies are very well suited as an interface between application developers and quantum computers because agents are inherently decentralized, decomposed, and autonomous by nature. Practical quantum computing involves hybrid solutions where some parts are solved by quantum computers and others by classical computers, making agent-based techniques ideal. Adding Quantum Agents expands agents' power to solve Constraint Satisfaction Problems in both the triggering and action components of agents. Quantum computing combined with agent-based technologies enhances both to a degree more than the sum of its parts.

References

1. Hayes-Roth, Barbara. "An architecture for adaptive intelligent systems." Artificial intelligence 72, no. 1-2 (1995): 329-365.
2. Jennings, Nicholas R. "On agent-based software engineering." Artificial intelligence 117, no. 2 (2000): 277-296.
3. DWave What is Quantum Annealing? https://docs.dwavesys.com/docs/latest/c_gs_2.html, last accessed 2021/11/24
4. Aaronson, Scott. "The limits of quantum." Scientific American 298, no. 3 (2008): 62-69.
5. Fuller, Tammy R., and Gerald E. Deane. "IoT applications in an adaptive intelligent system with responsive anomaly detection." In 2016 Future Technologies Conference (FTC), pp. 754-762. IEEE, 2016.
6. Fuller, Tammy R., and Gerald E. Deane. "Containerized Autonomous Agents for Cognitive and IoT Applications." Accepted IntelliSys, 5-6 September 2019 – London, United Kingdom, IEEE, 2019.
7. King, Andrew D., Jack Raymond, Trevor Lanting, Sergei V. Isakov, Masoud Mohseni, Gabriel Poulin-Lamarre, Sara Ejtemaee et al. "Scaling advantage over path-integral Monte Carlo in quantum simulation of geometrically frustrated magnets." Nature communications 12, no. 1 (2021): 1-6.
8. Sharabiani, Mansour TA, Vibe B. Jakobsen, Martin Jeppesen, and Alireza S. Mahani. "Quantum Computing in Green Energy Production." arXiv preprint arXiv:2105.11322 (2021).
9. Okada, Shuntaro, Masayuki Ohzeki, Masayoshi Terabe, and Shinichiro Taguchi. "Improving solutions by embedding larger subproblems in a D-Wave quantum annealer." Scientific reports 9, no. 1 (2019): 1-10.