Applied Computer Science, vol. 19, no. 2, pp. 43–54
doi: 10.35784/acs-2023-13

Submitted: 2023-04-29 | Revised: 2023-05-26 | Accepted: 2023-05-29

Keywords: Real-time task scheduling, Genetic algorithms, Concurrent computing

Mariano LARIOS-GÓMEZ [000-0002-2089-0608][*],[**],
Perfecto M. QUINTERO-FLORES [0000-0001-7651-4364][*],
Mario ANZURES-GARCÍA [0000-0001-6138-322][**],
Miguel CAMACHO-HERNANDEZ [0009-0002-8627-9876][**]

APPLICATION OF REAL-TIME FAN SCHEDULING IN EXPLORATION-EXPLOITATION TO OPTIMIZE MINIMUM FUNCTION OBJECTIVES

Abstract

This paper presents the application of a task scheduling algorithm called Fan based on artificial intelligence technique such as genetic algorithms for the problem of finding minima in objective functions, where equations are predefined to measure the return on investment. This work combines the methodologies of population exploration and exploitation. Results with good aptitudes are obtained until a better learning based on non-termination conditions is found, until the individual provides a better predisposition, adhering to the established constraints, exhausting all possible options and satisfying the stopping condition. A real-time task planning algorithm was applied based on consensus techniques. A software tool was developed, and the scheduler called FAN was adapted that contemplates the execution of periodic, aperiodic, and sporadic tasks focused on controlled environments, considering that strict time restrictions are met. In the first phase of the work, it is shown how convergence precipitates to an evolution. This is done in a few iterations. In the second stage, exploitation was improved, giving the algorithm a better performance in convergence and feasibility. As a result, a population was used and iterations were applied with a fan algorithm and better predisposition was obtained, which occurs in asynchronous processes while scheduling in real time.

1. INTRODUCTION AND MOTIVATION

Genetic algorithms were proposed as an alternative to solve problems that are difficult to solve in linear programming (Sreedhar, et al., 2020). In this work, a software tool was developed where genetic, heuristic, metaheuristic, and bioinspired algorithms were implemented and tested. One of the implemented algorithms was the Differential Evolution Algorithm (DEA) (Storn and Price, 1997; Cheng and Hwang, 2001), while others included the Harmony Search (HS) (Geem and Loganathan, 2001), the Whale Optimization Algorithm (WOA) (Nasiri and Khiyabani, 2018; Seyedali and Andrew 2016), among others.

[*] Universidad Autónoma de Tlaxcala (Facultad de Ciencias Básicas, Ingeniería y Tecnología), México
[**] Benemérita Universidad Autónoma de Puebla (Facultad de Ciencias de la Computación), México, mariano.larios@correo.buap.mx

Additionally, a genetic algorithm was implemented with the adaptation of the fan task planner proposed in Larios et al. (2019), which describes the task scheduling problem with processes and is expressed in the following point:

- Maintaining a communication path between nodes $v_i \rightarrow v_j$ is very complicated, considering that $i \neq j$ and v_j, $v_j \in V$. Where V is the set of nodes in a mobile network, which can be in a neighborhood L (Bertuccelli et al., 2010; Lim et al., 2016; Jeong, Simeone and Kang 2017; Wu and Zhang, 2018; Kim, Jung, Min, and Heo, 2021).
- The decentralization of objects and services is complex, i.e., by creating an object space between drones to obtain a mobile distributed environment (Soria, Schiano, and Floreano, 2021; Nouiri et al., 2018).
- The organization of the MDS is instable by not obtaining the best candidate as coordinator or leader (Saffre et al., 2022; Ramasubramanian, Haas and Sirer, 2003), this due is to the limited time in communication.

First, the genetic algorithm (GA) is presented as follows: given an initial population $X = (x_1, x_2,\ldots,x_n)$, where each individual's attributes from the i-th to the n-th element are different, ensuring that no individual repeats.

The generation of the population is in accordance with the Upper and Lower ranges marked as Li, Hi, with Li in the Lower set, Hi in Upper, in addition to Li, Hi and xi with i = 0, n attributes. The goal is to obtain a minimum element in the population using mutation, cloning, and genetic alteration operations where applicable.

The authors propose the implementation of a genetic algorithm by selecting the best fitness values, applying the real-time fan-out task scheduling algorithm (APTTRA), which helped to select tasks that can achieve better results based on the search for better fitness values. The communication required in one or more neighborhoods is within a set of processes that must be performed using a dynamic topology, which presents the problems established in Larios et al., (2019).

Section 2 presents the function used in the HSW and the results in the work of Portilla (Portilla et al., 2017; Barbosa et al., 2019), with the indicated constraints and respecting the Lower and Upper range specifications. The application of this algorithm provides asynchronous acceleration of results, while ensuring the timing of the task.

Finally, section 3 highlights the results of the proposed function applications and shows the results in the HSW and the Java application of the functions. The analysis and development of the proposed algorithm implemented in the HSW are also discussed. Additionally, the results with the convergence and feasibility graphs based on the exploration-exploitation of a given population are presented.

2. PROBLEM FORMULATION AND ANALYSIS

The authors' goal is to obtain a minimal element in the population using mutation, cloning and genetic alteration surgery, if necessary. Communication required between one or more neighborhoods within a set of processes must be performed using dynamic topology. It starts with the application of the functions:

$$f_1(\vec{X}\,) = (x_1 - 10)^3 + (x_2 - 20)^3 \tag{1}$$

with restrictions:

$$g_1(\vec{X}) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0 \qquad (2)$$
$$g_2(\vec{X}) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leq 0$$

The optimal solution established in Portilla et al. (2017) is in equation (1), Lower and Upper thresholds are applied, resulting in x = -6961.81387558, where x = (14.095000000000000064, 0.8429607892154795668), and the applied constraints are g1(x) and g2(x) (see equation 2).

In the software tool, the population with Upper and Lower ranges was created. With this tool, the population with its dimensions and ranges can be represented by taking each element of the matrix with dimensions by attributes for each $x_{i,j}$. The attributes of the individuals such as the i-th values and the dimension of the population as the j-th values are randomly selected, only if $L_i \leq x_{i,j} \leq H_i$ in a defined range.

These candidate individuals are sorted so that the value of the objective function corresponding to the first solution vector is minimized. In other words, feasible solutions are sorted in descending order according to their objective function value. However, it should be mentioned that the only feasible elements are those that satisfy the constraints established by the algorithm and it is inserted into a vector called Aptitude, where unfeasible candidates are discarded from the process.

$$\left(x_1^1 \ \cdots \ x_n^1, Fac_g \ \vdots \ddots \vdots \ x_1^{Dim} \ \cdots \ x_n^{Dim}, Fac_g \ \right) \qquad (3)$$

Having the initial population matrix (see equation 3), where $[x_{1i}, x_{2i}, \ldots, x_{ni}]$ (i = 1, 2, …, g) is a candidate solution in the g-th generation, Fac_g is also represented as an indicator of the number of constraint violations. In the creation of a population, everyone is evaluated. In case it violates the constraints of functions g1 and g2 as a set V, the condition shown in algorithm 1 is satisfied: v_1, v_2, v_t in *V* are taken to show feasibility for everyone in the initial population.

Algorithm 1. Keep the restrictions g_1 y g_2

```
1: People growth $X = (x_1, x_2, \ldots, x_n)$;
2: $v_1 = g_1(\vec{X}) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100$;
3: $v_2 = g_2(\vec{X}) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81$;
4: if $v_1 \leq 0$ and $v_2 \leq 0$ then
5:      $v_t=0$;
6: else
7:   if $0 \leq v_1$ and $v_2 \leq 0$
8:      $v_t=v_1$;
9:   else
10:    if $v_1 \leq 0$ and $0 \leq v_2$ then
11:       $v_t=v_2$;
12:    else
13:       if $0 \leq v_1$ and $0 \leq v_2$ then
14:          $v_t=v_1+v_2$;
15:       end if
16:     end if
17:   end if
18: end if
```

3. METRIC DESIGN AND APPLICATION

The algorithm of Fan used for the application of the metric (see Fig. 1), highlights the management, creation, and escalation of processes within a specific time. The processes fulfill the task of sending messages at a certain moment, creating a process per message, in this way the metric is obtained as a function of time. Each process was identified by an identifier, if the value is different from zero, then the creation of child processes is extended, and task scheduling is improved with the fan algorithm. If this is not the case, the processes are suspended.

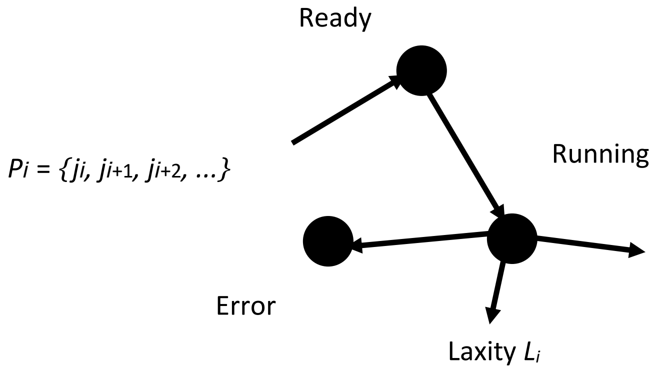

Fig. 1. Creation and scaling of processes with a specific deadline

In the planning of each process, they are identified as ready processes with a set of J_n tasks, then the creation of P_m processes is extended, avoiding entering the error state (E), if the deadline is met.

In equation (4), we have the following condition: if the slack time (lost time) X_j of each process is greater than the sum of the absolute constrained times d_i, then the process will be on hold.

$$E = \sum_{i=1}^{n}(d_i) \leq X_j \tag{4}$$

where: X_j – time which the task can be delayed in its activations to complete its deadline, d_i – absolute deadline.

It is expected to avoid the planning problem that is the domino effect, by defining a real-time scheduler σ(t) whose processes are generated in a spanning tree, such as a set of processes P, a set of resources R, and a set of tasks J, as shown in the equation (5).

Supposing that the solution is using a related acyclic graph reduces these planning errors, as is proposed in Larios-Gómez et al., (2019), where this work was the main source for the article.

$$G < P, R, J, A > \tag{5}$$

4. APTTRA APPLICATION IN A GENETIC ALGORITHM

When applying the genetic algorithm, an initial population is given in a lower range L and upper range H. In this algorithm, a stage of selection of an individual from the population called Ip is proposed. In this stage, Deb's rules (Deb, 2000) are applied, with the condition of the smallest number of violations with respect to the constraints g1(x), g2(x).

Algorithm 2. The APPTRA (Adaptive Penalty PArticle Swarm Optimization with TRimming and Aging) is applied in the genetic algorithm.

```
People creation $X = (x_1, x_2, \dots, x_n)$;
Range creation $L_i \in Lower$ y $H_i \in Upper$;
for $g = 0 \rightarrow iteracciones$ do
    $N_x$=clona;
  while $v_{1Nx} \le 0$ and $v_{2Nx} \le 0$ do
      selección de $N_x$;
  end while
  for $I_p \rightarrow Población$ do
    if $v_{1Nx} \le 0$ and $v_{2Ip} \le 0$ then
      sustituir $I_p = N_x$ ;
    end if
    if $v_{1Nx} \le 0$ and $0 \le v_{2Ip}$ then
      sustituir $I_p = N_x$;
    else if
    if $0 \le v_{1Nx}$ and $0 \le v_{2Ip}$ and $Nx_{apt} < Ip_{apt}$ then
      substitute $I_p = N_x$;
    end if
    Fit = min $(Nx_{apt}, Ip_{apt})$;
  end for
  $VIP_x$(Fit);
end for
```

In algorithm 2, there is a cloning stage of an agent called N_X, which is compared to a previously selected individual. The clone N_X is genetically modified, that is, it must comply with the ranges L_i and H_i, which will be bounded depending on the solutions in several generations or iteration that is equal to a percentage in which the simple comparisons are made. This proposal makes it possible to apply exploitation in the initial population, changing the way in which N_X and Ip are compared. After bounding the ranges L and H, new generations are improved by combining individuals with modified clones within those ranges. In the same way, cloning is improved in a percentage of generations due to iterations. Before ending the algorithm, it sends the best solutions or fitnesses as a subset of the population called VIP_X. First, the ranges are bounded depending on the best fitnesses sent in the vector Fit(x) as shown in algorithm 3.

Algorithms 2 and 3 are dedicated to exploiting the subpopulation, obtaining the best results from the initial population, which is approximately 10%. These individuals form a special subset called the VIP population. The process they undergo is drastic and fully exploitative, using a task scheduling to improve solutions to the limit. APTTRA is applied concurrently so that each thread is responsible for finding an optimal solution.

Algorithm 3. Automatic Adjustment of RRMa and RRMe Ranges

```
1: Lower Random Range $RRM_e$=-1;
2:  Higher Random Range  $RRM_a$=-1;
3: for $a = 0 \rightarrow attributes$ do
4:     if Lower ≤ $Fit_a$ + RRMe then
5:         $L_a$ = $Fit_a$ + $RRM_e$;
6:     else
7:         $L_a$ =Lower+$\epsilon$;
8:     end if
9: end for
10: for $a = 0 \rightarrow atributos$ do
11:     if Upper ≤ $Fit_a$ + RRMa then
12:        $H_a$ = $Fit_a$ + $RRM_a$;
13:     else
14:         Ha = Upper + $\epsilon$;
15:     end if
16: end for
```

5. RESULT

The software tool was applied to function (1) when only exploring and not enforcing constraints (2). The tool produced better results, but we reiterate that constraints were not applied at this time. Thanks to this tool, the genetic algorithm can be tested, and its convergence and feasibility graphs can be displayed.

The parameter values are:

– Dimension D = [50 to 100],
– Attributes = 2,
– Iterations G = [200 to 500],
– Exploration within the range [1-,1].

Constraints are applied, and exploration-exploitation is performed on a subpopulation with better fitness. Convergence with exploitation is shown in Fig. 1, where the hybrid work is evident after more than 900 iterations with a population of 100 solutions or individuals.

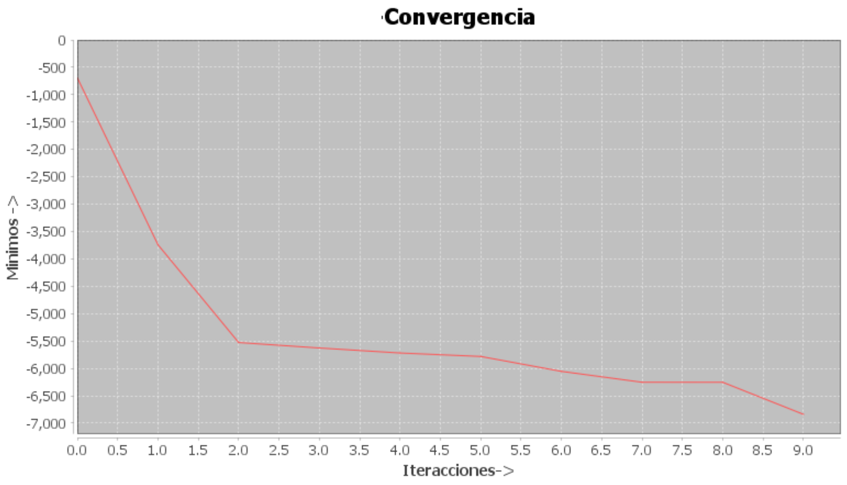

Fig. 1. Exploration-Exploitation Convergence of Function (1) with Constraints

The exploitation part provides us with a more complete result and if exploitation is not applied to the population, then the desired result will not be achieved. Because it is in this part where an algorithm must be applied that allows searching through intervals and genetic alteration or substitution, that is, some modification is applied in the creation of individuals, in this case it is range adaptation through the properties RRMa and RRMe applied in algorithm 3.

In the feasibility part, we worked in the same way as convergence, through exploration and exploration-exploitation.

Taking the parameter values as:

– Dimension D = [50 to 100],
– Attributes = 2,
– Iterations G = [200 to 500],
– Exploration-Exploitation,
– Range between [1-,1].

Restrictions are applied and forced exploitation is given to the subpopulation.

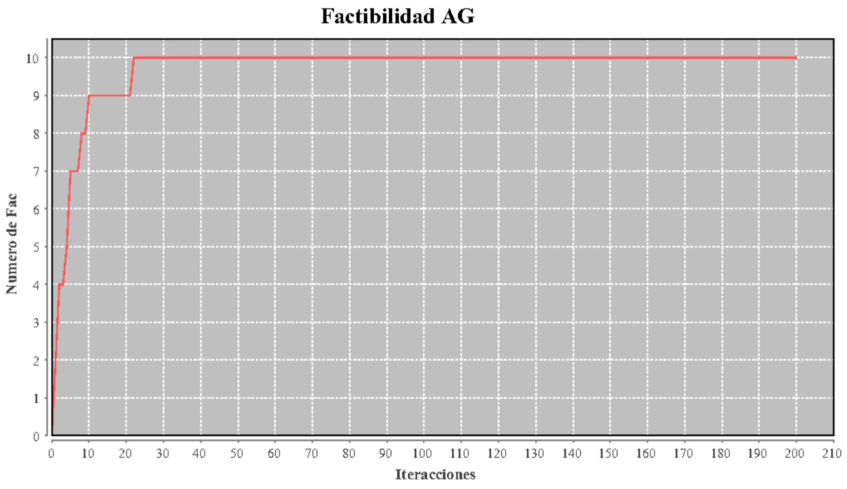

Fig. 2. Feasibility with Exploration-Exploitation with Constraints

The Exploration-Exploitation approach with APTTRA. Applying the restrictions and the hybrid Exploration-Exploitation approach, we have different results starting by applying Function (1) and restrictions with the attributes taken from the work of Portilla (Portilla et al., 2017; Barbosa et al., 2019): X = {14.09500000000000064,0.8429607892154795668} with several iterations Iter = 1000 and a population of 1000 individuals, the HSW gives us the result f(x) = -6961.813875580138 with $g_1(x) \leq 0.0$, $g_2(x) \leq 0.0$

The results we have so far do not improve upon those obtained in Function (1), even when applying the hybrid method proposed in the GA. However, we do have better fitness values, so this algorithm still needs to be further improved, detailed, and tested with more populations.

For example, we have the fitness in Figure 3, where better fitness values are obtained in a short time, if the APTTRA algorithm is applied.

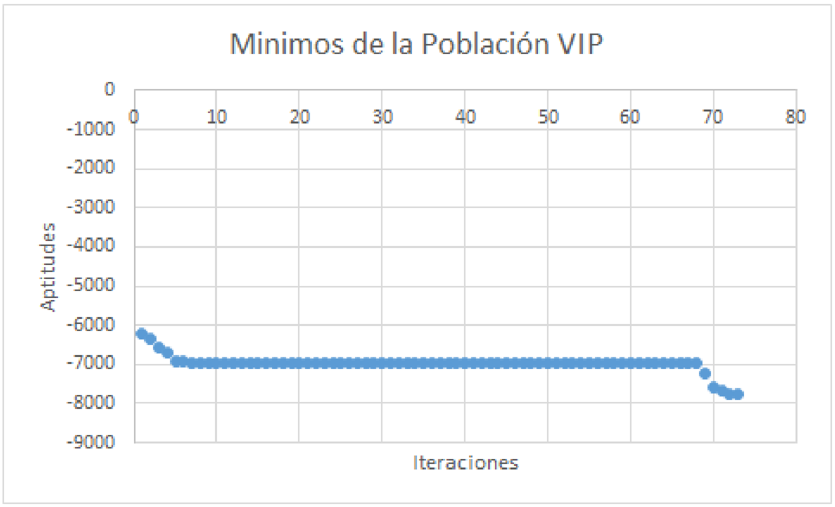

Fig. 3. Convergence of GA with exploration of the VIP population

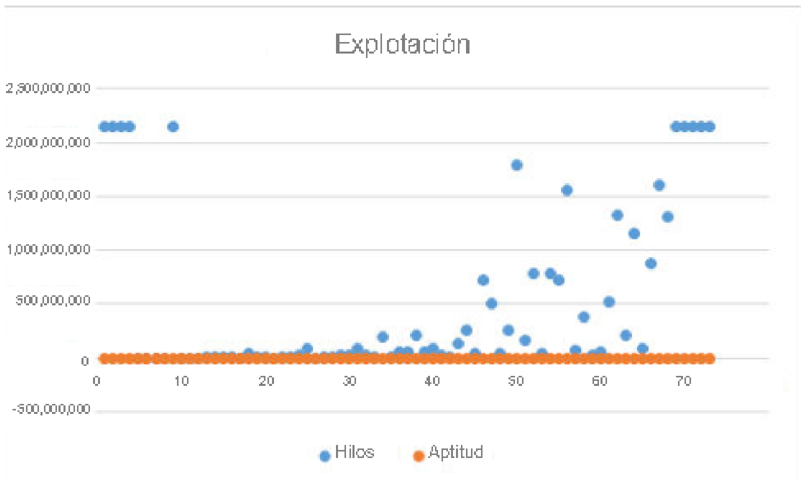

Fig. 4. AG with Exploration-APTTRA

Maximum iteration number for a search process: 2147483647.
Aptitude:

– x = {13.25345222386948, 0.15710759348131223},
– $g_1(\vec{x})$ = -8.42691952718755 Fulfills,
– $g_2(\vec{x})$ = -6.743823974926528 Fulfills.

Another function where AG with APTTRA was applied is equation (6) with constraints and attributes obtained from the work of Portilla (Portilla et al., 2017; Barbosa et al., 2019). The result was x = {2.17199634142692, 2.3636830416034, 8.77392573913157, 5.09598443745173, 0.990654756560493, 1.43057392853463, 1.32164415364306, 9.82872576524495, 8.2800915887356, 8.3759266477347} with a maximum iteration number Iter = 2000 and a population of 10 individuals.

$$f_4(\vec{x}) = x_1^2 + x_2^2 + x_1\,x_2 - 14\,x_1 - 16\,x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + \\ +(x_5 - 3)2 + 2(x_6 - 1)^2 + 5\,x_7^2 + 7(x_8 - 11)^2 + \\ +2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45 \tag{6}$$

With restrictions

$$\begin{gathered} g_1(\vec{x}) = -105 + 4\,x_1 + 5\,x_2 - 3\,x_7 + 9\,x_8 \ \le 0 \\ g_2(\vec{x}) = 10\,x_1 - 8\,x_2 - 17\,x_7 - 2\,x_8 \ \le\ 0 \\ g_3(\vec{x}) = -8\,x_1 + 2\,x_2 + 5\,x_9 - 2\,x_{10} - 12 \le 0 \\ g_4(\vec{x}) = 3(x_1 - 2)^2 + 4(x_2 - 3) + 2\,x_3^2 - 7\,x_4 - 120 \le\ 0 \\ g_5(\vec{x}) = 5\,x_1^2 + 8\,x_2 + (x_3 - 6)^2 - 2\,x_4 - 40 \le\ 0 \\ g_6(\vec{x}) = x_1^2 + 2(x_2 - 2)^2 - 2\,x_1\,x_2 + 14\,x_5 - 6\,x_6 \le\ 0 \\ g_7(\vec{x}) = 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3\,x_5^2 - x_6 - 30 \le\ 0 \\ g_8(\vec{x}) = -3\,x_1 + 6\,x_2 + 12(x_9 - 8)^2 - 7\,x_{10} \le\ 0 \end{gathered} \tag{7}$$

The HSW yields the result f(x) = 24.30620906817991 with g1(x), g2(x) less than zero, compared to the results in Portilla of 24.30620906818. It is mentioned that the recorded results may suffer from rounding errors that can cause slight infeasibility sometimes in the best solutions given with six active constraints g1, g2, g3, g4, g5, and g6, and in this work, eight active constraints are achieved.

Search process iteration number: 104531.

Fitness: 24.30620906817991 Meets requirement.

Attributes x = {2.1855839721698693, 2.309316372623711, 8.625018811507283, 5.0471897772421155, 0.8678245864139478, 1.2978045356201755, 1.0039750892086519, 9.670233173002996, 8.177772543738563, 8.190078280631027}

$g_1(\vec{x})$	= -0.6909089588009465	Fulfills
$g_2(\vec{x})$	= -33.02673412184407	Fulfills
$g_3(\vec{x})$	= -0.35733287468077535	Fulfills
$g_4(\vec{x})$	= -9.20783972031191	Fulfills
$g_5(\vec{x})$	= -0.8452383156983103	Fulfills
$g_6(\vec{x})$	= -0.7635621691254668	Fulfills
$g_7(\vec{x})$	= -6.4179068691661065	Fulfills
$g_8(\vec{x})$	= -49.652164717497186	Fulfills

```
#include <sys/socket.h>
#include <sys/time.h>
#include <arpa/inet.h>
#include <unistd.h>
struct sockaddr_in serverSocketAddr;
struct sockaddr_in clientSocketAddr;
typedef struct{
    char * message;
    char * ip;
    int port;
}ToNodo;
void sendToMessage(char * message, char * ip, int port){
    struct sockaddr_in toAddress;
    toAddress.sin_family = AF_INET;
    toAddress.sin_addr.s_addr = inet_addr(ip);
    toAddress.sin_port = htons(port);
    bzero( &(toAddress.sin_zero), 8);
    sendto(idServerSocket, message, sizeof(message), 0, (struct sockaddr*)&toAddress, sizeof(toAddress));
}

void receiveToMessage(char * message){
    int clientSocketAddrLen = sizeof(clientSocketAddr);
    int receive = recvfrom(idServerSocket, message, sizeof(message), 0, (struct sockaddr *)&clientSocketAddr, &clientSocketAddrLen);
}
```

Fig. 5. Methods for the realization of a P2P network in a dynamic distributed mobile environment

To develop an aerial mobile distributed system applying a network connection based on p2p nodes, where each peer is represented as a distributed mobile device (see Fig 4). Taking this into account, novel real-time scheduling algorithms were designed and implemented to observe the delay quality in a real use case. For example, it is contemplated to design and implement an embedded system with drones to report the help of people in environments that are difficult to access or in a natural disaster.

6. CONCLUSIONS AND FUTURE WORKS

In this paper, the application of a task scheduling algorithm called Fan was presented; where an algorithmic approach for the field of artificial intelligence adjusts to the ranges of the best fitness values through automatic random range changes and task planning exploitation, seeking a solution until the resource is exhausted.

This approach was applied to planning algorithms such as APTTRA, leading to improved results overall. Furthermore, techniques, methods, and algorithms of distributed mobile systems were studied; therefore, a methodology and a metric capable of satisfying the objectives and the goal in the distributed environment were proposed. Finally, a proposal for future work is to decentralize the planning work in a hybrid Exploration-Exploitation way and create processes called Miners, which would provide a Hard-Exploitation, synchronized with APTTRA.

Acknowledgments

The authors acknowledge the people from the National Laboratory of Supercomputing of Southeast Mexico which belongs to the CONACyT national laboratories, for all the technical assistance and the computational resources.

REFERENCES

Bertuccelli, L. F., Beckers, N.W. M., & Cummings, M. L. (2010). Developing operator models for UAV search scheduling. In *AIAA Guidance, Navigation, and Control Conference* (p. 7863). American Institute of Aeronautics and Astronautics.

Cheng, S. L., & Hwang, C. (2001). Optimal approximation of linear systems by a differential evolution algorithm. *IEEE Transactions on Systems, man, and cybernetics-part a: systems and humans, 31*(6), 698-707. https://doi.org/10.1109/3468.983425

Deb, K. (2000). An efficient constraint handling method for genetic algorithms. Computer methods in applied mechanics and engineering, 186(2-4), 311-338.

Geem, Z. W., Kim, J. H., & Loganathan, G. V. (2001). A new heuristic optimization algorithm: harmony search. *Simulation*, 76(2), 60-68. https://doi.org/10.1177/003754970107600201

Jeong, S., Simeone, O., & Kang, J. (2017). Mobile edge computing via a UAV-mounted cloudlet: Optimization of bit allocation and path planning. *IEEE Transactions on Vehicular Technology*, 67(3), 2049-2063.

Kim, B., Jung, J., Min, H., & Heo, J. (2021). Energy efficient and real-time remote sensing in AI-powered drone. Mobile Information Systems, 2021.

Larios-Gómez, M., Carrera, J. M., Anzures-García, M., Aldama-Díaz, A., & Trinidad-García, G. (2019). A Scheduling Algorithm for a Platform in Real Time. In International Conference on Supercomputing in Mexico (pp. 3-10). Springer, Cham.

Lim, G. J., Kim, S., Cho, J., Gong, Y., & Khodaei, A. (2016). Multi-UAV pre-positioning and routing for power network damage assessment. IEEE Transactions on Smart Grid, 9(4), 3643-3651.

Nasiri, J., & Khiyabani, F. M. (2018). A whale optimization algorithm (WOA) approach for clustering. Cogent Mathematics & Statistics, 5(1), 1483565.

Nouiri, M., Bekrar, A., Jemai, A., Niar, S., & Ammari, A. C. (2018). An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem. Journal of Intelligent Manufacturing, 29(3), 603-615.

Barbosa-Mendez, M. A., Portilla-Flores, E. A., Vega-Alvarado, E., Calva-Yáñez, M. B., & Sepúlveda-Cervantes, G. (2019, September). A harmony search variant based on a novel synthesized approach for constrained numerical optimization. In 2019 16th international conference on electrical engineering, computing science and automatic control (CCE) (pp. 1-6). IEEE.

Portilla-Flores, E. A., Sánchez-Márquez, Á., Flores-Pulido, L., Vega-Alvarado, E., Yáñez, M. B. C., Aponte-Rodríguez, J. A., & Niño-Suárez, P. A. (2017). Enhancing the harmony search algorithm performance on constrained numerical optimization. IEEE Access, 5, 25759-25780.

Ramasubramanian, V., Haas, Z. J., & Sirer, E. G. (2003, June). SHARP: A hybrid adaptive routing protocol for mobile ad hoc networks. In Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing (pp. 303-314).

Saffre, F., Hildmann, H., Karvonen, H., & Lind, T. (2022). Self-swarming for multi-robot systems deployed for situational awareness. In New Developments and Environmental Applications of Drones (pp. 51-72). Springer, Cham.

Seyedali, M., & Andrew, L. (2016). The Whale Optimization Algorithm Advances in Engineering Software.

Soria, E., Schiano, F., & Floreano, D. (2021). Distributed Predictive Drone Swarms in Cluttered Environments. IEEE Robotics and Automation Letters, 7(1), 73-80.

Sreedhar, M., Reddy, S. A. N., Chakra, S. A., Kumar, T. S., Reddy, S. S., & Kumar, B. V. (2020). A review on advanced optimization algorithms in multidisciplinary applications. Recent Trends in Mechanical Engineering: Select Proceedings of ICIME 2019, 745-755.

Storn, R., & Price, K. (1997). Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization, 11(4), 341.

Wu, Q., & Zhang, R. (2018). Common throughput maximization in UAV-enabled OFDMA systems with delay consideration. IEEE Transactions on Communications, 66(12), 6614-6627.