Self-adaptive agents based on reinforcement learning to optimize patient scheduling in emergency department

Faiza Ajmi, Sarah Ben Othman, Faten Ajmi, Hayfa Zgaya-Biau, Jean-Marie Renard, Gregoire Smith, Slim Hammadi

Self-adaptive agents based on reinforcement learning to optimize patient scheduling in emergency department

Faiza Ajmi
FGES, Université Catholique de Lille F-59000
Lille, France
faiza.ajmi@univ-catholille.fr

Sarah Ben Othman
CRIStAL CNRS UMR 9189
Villeneuve d'Ascq, France
sara.ben-othman@centrelelille.fr

Faten Ajmi
CRIStAL CNRS UMR 9189
Villeneuve d'Ascq, France
fatenajmii@gmail.com

Hayfa Zgaya Biau
CRIStAL CNRS UMR 9189
Villeneuve d'Ascq, France
hayfa.zgaya-biau@univ-lille.fr

Jean-Marie Renard
Emergency dotor RHC Lille
Oscar Lambret, France
jean-marie.renard@univ-lille.fr

Gregoire Smith
Emergency dotor RHC Lille
Oscar Lambret, France
gregoire.smith@univ-lille.fr

Slim Hammadi
CRIStAL CNRS UMR 9189
Villeneuve d'Ascq, France
slim.hammadi@centrelelille.fr

***Abstract*—In this article, we present a framework ABOS based on a collaborative multi-agent system (MAS) for multi-skill health care scheduling using metaheuristics. In this framework, each agent preforms his actions autonomously in the search space of a scheduling optimization problem. Information about patient scheduling is shared between agents who collaborate through the dynamic environment. The objective is to allow the agents to adapt their decisions using the Reinforcement Learning approach according to the acquired experience with the interaction with the other agents and the environment. The aim of this interaction between agents is to enhance the quality of the solutions provided by the agents from the search space. Experiments were performed using real data provided by the adult emergency department (AED) of Lille university hospital center (LUHC). The simulation results confirm that the integration of Machine Learning in agent behaviors impacts the quality of the scheduling solution. Collaboration between agents in "friend" or "enemy" mode influences the quality of the solution as well and thus impacts the health care patient pathway.**

***Index Terms*—Collaborative optimization, Multi-agent system, Reinforcement Learning, Q-Learning, Hybrid metaheuristics**

I. Introduction

Emergency departments (EDs) are often the first point of admission to provide urgent care to urgent patients without a prior appointment [1]. As a result, ED operations are more challenging and resources are limited due to diverse patient needs, different levels of treatment, and unexpected patient arrival times that differ from other hospital departments. The random nature of patient arrivals leads to malfunctions, overcrowding situations, increased waiting times, increased length of stay, increased medical errors, and increased patient mortality. Indeed, new health care needs have emerged [2], and patients are looking for a fast and better service. In addition, reducing patients' treatment and waiting timesfto be perfor can minimize financial losses because the faster patients get out of the service, the more patients a given physician can see. Because EDs are often concerned with overcrowding problem and limited human and material resources, optimizing health care tasks scheduling can be a key solution to address this problem [3].

The management of hospital systems in general involves three issues: their design, planning and control. Design is the definition/prediction of the future characteristics of the hospital system. In this phase, questions about the design of the system, the human and material resources used, or the order in which patients receive treatment are addressed. Planning relates to the determination of the different resources used by the health care operations to be performed on patients and to define how and when these operations are to be performed. Finally, control is concerned with correcting any inherent disruptions. These disruptions may be a worsening of a patient's state of health, new patients to be treated, and therefore new care tasks to be carried out, or more generally, any random event that impairs the achievement of the health care treatment plan determined by the scheduling. These three key steps are executed according to the frequency of problems encountered during the execution of the scheduling. In order to optimize the management of EDs, the optimization approach must provide methods that can be adapted to the dynamic aspect of the hospital system and that can improve the quality of a solution. Metaheuristics allow to easily define methods with a strong power of intensification and provide good quality solutions rapidly. Some of the previous works in the literature [4] use metaheuristics to solve hospital scheduling problems. Patients' waiting time can be significantly reduced through providing a better workplan. In our previous work, an ABOS framework (Agent-Based Optimization Systems) based on a reactive MAS was proposed in [5]. This system is characterized by a distributed control between several entities called reactive agents. The idea in this paper is to set up a cooperative process

This work is supported and financed by the National Research Agency.

to coordinate the performance of metaheuristics, that is, the genetic algorithm (GA), the simulated annealing algorithm (SAA) and the taboo algorithm (TA). Each agent is charged to explore the advantages of a metaheuristic. The overall performance of the system is evaluated from a global utility function defined a priori and characteristic of the problem to be solved. It is then up to the agents to exhibit individual behaviors in such a way that the fitting of these behaviors at runtime succeeds in developing a collective behavior that optimizes this global utility function. Indeed, agents have only partial perceptions of their environment: they cannot directly access the global utility function and only have a local perception of the solutions found. The agents must then cooperate and adapt during the execution of the system to take into account possible new perceptions and the evolving behaviors of the other agents. This cooperation is based on two criteria to make the best use of metaheuristics (the details of this cooperation are published in our previous work [6]) :

- Competition (enemy mode): In order to achieve effective cooperation, it is necessary for the process to be driven by a competition between metaheuristics. This competition is established through the calculation of the system's performance indicators. This evaluation helps to choose the metaheuristics to be carried out at each decision point.
- Collaboration (friend mode): In order to enhance the quality of the solutions, it is useful to associate the concept of competition with that of collaboration between the resolution methods. This is implemented by the mechanism of communication of the solutions found. Indeed, at the end of the execution of each metaheuristic, the collaboration between agents system must save the best solution found in order to communicate it to the next metaheuristic to be executed, allowing this latter to be more efficient.

This is a complex problem because the control of the system is carried out at the individual level whereas its evaluation is carried out at the collective level. We wish to take advantage of the multiplicity of agents in order to bring out collective behaviors that are qualitatively different from individual behaviors. When evaluating metaheurisctics, a Reinforcement Learning algorithm is used to carry out the task of choosing optimal solutions in the search space. This algorithm aims to optimize output parameters which are the step of fitness optimisation for the input of metaheuristics and generate a good value for optimisation at the output. We propose in this work entirely decentralized learning techniques that allow the agents to automatically adapt their behavior to the collective task to be solved without having a global vision of the system. As we wish to make decentralized learning, each agent must have a good evaluation of the long-term consequences of a decision, in particular with regard to the impacts of his actions on the other agents. Several methods using machine learning techniques have been developed to optimize tasks scheduling. For example, a 2-stage strengthening training algorithm is defined in [7] for task planning. The authors propose an algorithm which uses a profoundly intensive learning phase to construct a deployable resource mapping strategy. A stochastic gradient descent based method is proposed in [8]. The authors modeled the average completion time decreasing problem. In the selection of the best strategy for task scheduling, authors in [9] extend the work of [10] and employ Monte Carlo Tree Search (MCTS). The effectiveness of the algorithm is proved. A framework based on a deep-neural networking programmer inspired by CNN was proposed in [11]. The authors use a two-dimensional convergence approach to plan tasks. Many researchers have focused on the use of Reinforcement Learning associated with metaheuristics to solve optimization problems [12] but it is to be noted that most of the proposals do not use Reinforcement Learning among their structures for solving optimization problems [13]. The originality of our work is to use conjointly metaheuristics, MAS and Reinforcement Learning for the sake of optimization. This work is part of OIILH project supported and financed by the National Research Agency and is performed in the AED of LUHC (Lille, France).

II. ABOS FRAMEWORK DESCRIPTION

The current article presents an improvement of our ABOS framework already published in [5] by using the multi-agent approach allowing the hybridization of metaheuristics and Reinforcement Learning to solve multi-skill health care tasks scheduling in ED. In [5], our framework presents a communication protocol between agents for solving combinatorial optimization problems. In this previous work, the communication protocol is a generic and evolutive structure, in which each metaheuristic is defined as an autonomous agent who interacts with each other following a predefined collaborative process. The main contributions of this paper are to:

- Develop the self-adaptive approach of the ABOS agents, integrating the Q-Learning algorithm in the behavior of the agents to better adapt their metaheuristics to solve the patient scheduling problem.
- Improve collaboration between agents by using a shared memory space (SMS) containing good solutions stored and improved by agents during the iterations ensuring a wide diversity of solutions.
- Show how the self-adaptive skills of agents allow to improve directly their performance to search optimal solutions.

A. The Multi-agent metaheuristic approach

The adaptive muti-agent metaheuristic optimization described in this paper is integrated as a main part of the ABOS framework. In this framework, each agent integrates in his own behavior a metaheuristic and has the objective of finding the solution for a given multi-skill health care tasks scheduling problem. The intelligence of ABOS agents depends on a coherent definition of their environment, a necessary condition for their autonomy. This environment is characterized by the search space of the addressed problem. Therefore during the

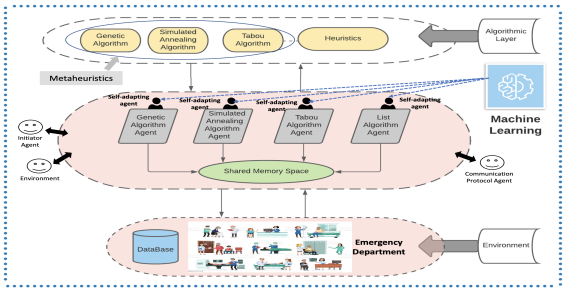


Fig. 1. ABOS architecture improvement

search process of the scheduling solution, the metaheuristic-agents in the ABOS framework should access to the global MAS environment. The new architecture of ABOS framework is defined in Fig.1.

Each agent has actions that define his global vision that he will have of the environment. He does not have a complete knowledge of the environment. The objective of ABOS is to use, at the same time, the strengths and advantages of each metaheuristic through the collaborative work of the agents. The architecture of ABOS is scalable because it will be able to add new optimizing agents without impacting the existing ones. These new agents are equipped with other algorithms and heuristics to enrich the environment and increase the diversity of SMS solutions. The ABOS agents interact with the environment by collaborating with other agents either as friends or as enemies (considering other agents as competitors). These interactions allow agents to exchange and share information about their state and the environment. The initial ABOS conceptual and development model is described in [5]. In this paper, we improve this framework by including the following MAS entities: (i) Environment; (ii) Initiator Agent; (iii) Metaheuristics Agents (Genetic agent, Simulated annealing agent, Tabou agent) and (iv) Communication protocol Agents. The collaborative optimization structure of ABOS architecture is improved in this paper. A SMS is used for exchanging information. The main objective of this ABOS improvement is the need to increase the autonomy and self-adaptation of the agents. The new structure of ABOS framework is composed of four principle steps which are presented in this paper:

i Environment: defined mainly by the search space of the studied problem. Therefore, it provides all information needed for solving the multi-skill health care tasks scheduling problem, for example the number of patients to be scheduled in the ED, the number of health care professionals in this department, the number of care rooms available, and so on;
ii SMS of Solutions: its main task is to provide a shared solutions for all agents at each iteration;
iii Metaheuristic Agents: in charge of monitoring the search for the good solutions;
iv Finally, new self-adaptive agents, incorporating the Reinforcement Learning approach.

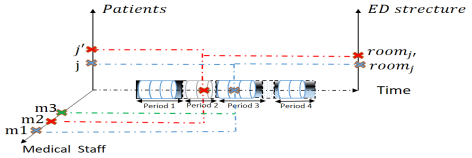


Fig. 2. Four-dimensional hypercube model

B. ABOS : model patient scheduling solution

The most appropriate way to facilitate the multi-skill health care tasks scheduling analysis and decision-making is a multidimensional solution modeling. The latter represents the solution as points in a multidimensional space. The subjects of analysis (Patient, Medical Staff, Time, Location) are studied along several axes (the dimensions). In ABOS the patient scheduling solution is modeled through four-dimensional (hypercube) whose axes are: *Medical staff*, *Patients*, *ED structure* and *Time* (Fig.2). The time axis is divided into intervals. Each interval has a different size. The time schedule is divided into several periods that do not necessarily have the same length. When two periods have the same length, the number of slots in each period may differ. In general, a period has several slots. Thanks to the division of time axis into many slots, each medical staff member is assigned to a patient in a specific slot belonging to a specific period. For example, as it is shown in Fig.2, medical staff $m2$ treats patient j' in the second slot of period 2 in operation $room_{j'}$. In the same figure, we have a parallel assignment of 2 medical staff members $m1$ and $m2$, respectively, to patients j and j' in the periods 2 and 3, in operation $room_j$ and operation $room_{j'}$. Fig.2 also describes the multi-skill hybercube assignment which is possible in our approach thanks to the choice of the chromosome which can be adapted to the problem treated. Here the patient j needs 2 different skills for his treatment, $m1$ and $m3$ assigned in the same period 3 and the same slot in operation $room_j$.

III. PATIENT SCHEDULING ENVIRONMENT

This section focuses on scheduling patients in the AED of LUHC according to the priority of patients' healthcare tasks determining by the triage process. This problem is similar to a Flexible Job Shop Scheduling problem with jobs are patient health care tasks, machines are human and material resources (physicians, boxes, beds, ...) and operating sequences are patients' pathways which depend on patients' pathologies. The ABOS is an adequate framework to solve conjointly the problems of patients scheduling and coordination in the emergency service. Solving these problems efficiently is crucial to improve the performance of the ED and insure a more reliable framework. In this way, we model the solution by a 4-dimentional hypercube whose axes are : Time axis, Patient axis, Medical Satff axis and Location axis. Time axis is divided in 5 minutes slots (Fig.2).

A. Notations

The main indices and decision variables of the patient scheduling are defined as follows:

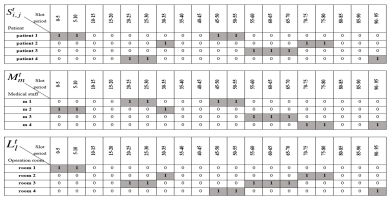


Fig. 3. Example of four patients scheduling solution

Indices

J : emergency patient index, $j \in \{1 \,..\, J\}$,
I : emergency care tasks index, $j \in \{1 \,..\, I\}$,
i, j : the care task i of patient j,
M : emergency medical staff index, $m \in \{1 \,..\, M\}$,
L : emergency operation room index, $l \in \{1 \,..\, L\}$,
T: slot period index, $t \in \{1 \,..\, T\}$, where the slot $T+1$ indicates the slot period outside the current scheduling horizon H,

Decision variables,

$S_{i,j}^t$: boolean, set to 1 if the care task i of patient j is scheduled in the slot period t and set to 0 otherwise,
M_m^t: boolean, set to 1 if the medical staff m is assigned to slot period t and set to 0 otherwise,
L_l^t: boolean, set to 1 if the operation room l is available in the slot period t and set to 0 otherwise,
$W_{H,j}$: the waiting time of scheduled patient j in horizon H
$t_{ar,j}$: the arrival time of patient j
$t_{fc,j}$: the first consultation time for the patient j

Based on these notations, the patient schedule is admissible if only checks the equation 1.

$$1 \leq \sum_{t=1}^{T} \sum_{j=1}^{n} S_{i,j}^t = \sum_{t=1}^{T} \sum_{l=1}^{card(B)} L_l^t \leq \sum_{t=1}^{T} \sum_{k=1}^{m} M_{k,j}^t \quad (1)$$

The equation above checks the feasibility of the scheduling by verifying that, in each time a care task i of a patient j is scheduled in the time slot t, we have at the same time an assigned medical staff m in a room l in the same slot t to perform this task. Let the waiting time of scheduled patient j (W) be the sum of patients' waiting time between the registration time and the first consultation time, where:

$$W = Min\ (\frac{\sum_{j=1}^{n}(max(0, t_{1,j} - ta_j) + \sum_{i=1}^{I_j} max(0, t_{i+1,j} - f_{i,j}))}{n}) \quad (2)$$

The objective is to minimize the sum of waiting time for all patient j scheduled in the horizon H. This is subject to the two types of constraints. The set of hard and flexible constraints presented in our previous work [1].

B. Patient scheduling neighborhoods

Many neighborhoods scheduling functions were developed and generated by the algorithm 1 to explore the search space of solutions in our ABOS framework instantiation. These neighborhood functions define the set of states integrated in the agent learning approach (presented in the section IV).
We denote NS an enumerate type containing four-neighborhood structure:

- $NS = (MID, MIS, SDMS, SSMS)$ four neighborhood structure;
- MID: multiple insertion in different medical staff;
- MIS: multiple insertion in the same medical staff;
- $SDMS$: swap between different medical staff;
- $SSMS$: swap between same medical staff.

Algorithm 1: generate_neighborhood_solutions (SS, N_{max}, H)

Input
SS: current solution respecting the hypercube model
N_{max}: maximum number of required neighborhood solutions
H: scheduling horizon
$NS = (MID, MIS, SDMS, SSMS)$
0utput
SNS : set of neighborhood solution with $card(SNS) = N_{max}$
begin
$N \leftarrow 0$; $SNS \leftarrow \emptyset$;
while $(N < N_{max})$ **do**
  $mode = random(NS)$;
  if $mode = MID$ **then**
    $SS_1 \leftarrow$ multiple insertion of care task (i, j) chosen randomly in SS from one medical staff m to another one
    update $S_{i,j}^t$, M_m^t, L_l^t ;
    check the equation 1
  else if $mode = MIS$ **then**
    $SS_1 \leftarrow$ multiple insertion of care task (i, j) chosen randomly in SS in the schedule of the same medical staff m
    update $S_{i,j}^t$, M_m^t, L_l^t ;
    check the equation 1
  end
  else if $mode = SDMS$ **then**
    $SS_1 \leftarrow$ exchange move of one care task (i, j) chosen randomly in SS of medical staff with an (i, j) from another medical staff
    update $S_{i,j}^t$, M_m^t, L_l^t ;
    check the equation 1
  end
  else if $mode = SSMS$ **then**
    $SS_1 \leftarrow$ exchange move of one care task (i, j) chosen randomly in SS of medical staff with another (i, j) of the same medical staff ;
    update $S_{i,j}^t$, M_m^t, L_l^t ;
    check the equation 1
  end
  $N \leftarrow N + 1$
  $SNS \leftarrow SNS \cup \{SS_1\}$
end
return SNS
end

Example: neighborhood functions generated by the algorithm 1

Fig.3 shows a scheduling solution of four patients. The treatment of patient 1 requires four care tasks and for each of 3 other patients only 3 care tasks are required. We have 4 medical resources (4 doctors) and 4 consultation rooms. We note that the scheduling in Fig.3 is admissible because it verifies the equation 1. In addition, we also note that when the care tasks are scheduled in successive slot periods, they are carried out in the same operation room with the same doctor.

After several executions of algorithm 1, we select the five neighborhood functions:

1) care task assignment to different medical staff members **(A)**: This function realizes the re-assignment of patient care task from one medical staff to another medical staff. For example, in Fig.3, the third care task of patient 3 who is assigned to doctor $m4$ can be re-assigned to doctor $m3$.
2) successive care tasks assignment to different medical staff members **(B)**: This function realizes the re-assignment of successive patient care tasks to another medical staff member. For example, in Fig.3, the three health care tasks

of patient 3 who has been already assigned to doctor $m3$ can be re-assigned to doctor $m2$.

3) care task insertion in the same medical staff member work schedule **(C)** : this neighborhood function performs the move of one care task to another position in the medical staff schedule. For example, in Fig.3, the first care task of patient 2 changes position from the period slot [30-35] to the period slot [20-25]. Therefore the period slot [30-35] of doctor $m2$ as well as room 2 are updated (period slot [20-25]).
4) swap two care tasks between different medical staff members **(D)**: this neighborhood function performs the swap of one care task from one medical staff member to another. For example, in Fig.3, the first two care tasks of patient 1 will be assigned to the medical staff member $m1$. Therefore the first two care tasks of patient 4 will be assigned to the medical staff $m2$, rooms change accordingly.
5) swap between the same medical staff member **(E)**: this neighborhood function performs the change of position of one or more care tasks in the work plan of the medical staff member. For example, in Fig.3, the last care task of patient 4 changes position from the period slot [90-95] to the period slot [80-85] with the same medical staff member $m4$.

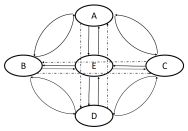
IV. SELF-ADAPTIVE AGENTS: REINFORCEMENT LEARNING

A. Shared memory space (SMS)

The main idea of the ABOS framework is to build agent metaheuristic architecture in order to produce new optimization schemes. Thanks to the collaboration and to the collective interaction structure, agents have the ability to guide their search in the solutions' space toward the most promising region, and thus, to improve the final results and reduce the computational time needed to solve the problem. The interaction between agents occurs through the exchange of information about the patient scheduling search space. The current scheduling solutions are stored in the SMS being available to be shared by the agents at the end of each iteration. The maximum size of the SMS is a parameter of the problem and the addition of new scheduling solutions is controlled by the algorithm 2, in order to regulate the SMS updating and maintain the diversity of solutions. The main goal of the algorithm 2 is to make the metaheuristic agents avoid premature convergence by diversifying as much as possible the solutions shared in the SMS. At the same time, the best existing (elite) solution in the SMS is always updated at each insertion, and its elimination is forbidden.

B. Self-adaptive agents

This section shows the self-adaptive agents approach using Reinforcement Learning, based on the Q-Learning algorithm, in order to improve the local search by the choice of an adequate neighborhood solutions. In the Reinforcement Learning model, there is no input/output data; the idea is that the agents take advantages of their experiences to improve their performances. Taking the advantage of discretization of the patient scheduling in a dynamic environment such as the ED. We can structure this problem into a Markov decision process (MDP). A Markov decision process framework is described by the tuple of (SS, AS, R, TP, γ), namely state space, action space, reward collection, transition probability matrix, and discount factor.



State/action	A	B	C	D	E
A	0	0	0	0	0
B	0	0	0	0	0
C	0	0	0	0	0
D	0	0	0	0	0
E	0	0	0	0	0

(a) A complete graph with five states (b) Initial Q-table used in Q-learning algorithm

Fig. 4. A complete states graph and Q-table

In this paper the MDP is defined as follows :

1) Set of States Space SS: states space are composed by the 5 neighborhood functions (subsection III-B), available for the patient scheduling problem to be handled by the ABOS framework. In the case of the test problems used here, we have: SS = {A, B, C, D, E };
2) Set of Action Space AS : an action is represented by an arc that connects two nodes. A node is represented by a state (neighborhood function). The set of actions can correspond to a complete graph whose vertices are adjacent two by two, i.e. any pair of disjoint vertices is connected by an arc. An example of a complete graph modeling the relationship between neighborhood functions (states) and actions is shown in Fig.4(a). the Q table corresponding to this example is shown in Fig.4(b). The dimensions of Q-table is in the form of square matrix with dimensions $(N * N)$, in which N is the number of states (the number of neighborhood functions of patient scheduling problem);
3) Reward R : based on the fitness value $reward(x)$ of the solution x and is evaluated on the basis of the objective function $f(x)$ of the problem.
4) Transition probability TP : for the probability of transition from a state s into a new state s' by an action a we use the control policy defined by $\Pi(s, a)$. This latter represents the probability of performing the action a in the state s. $\Pi(s, a)$ evolves according to the experience the agent acquires through interactions with the environment and with other agents. The learning process in the Reinforcement Learning is therefore expressed by the convergence to the optimal policy Π^*.
$$\Pi(s, a) = P_r(a_t = a \backslash s_t = s)$$
5) Discount factor γ : γ shows the importance of future reward to the current state.

C. Q-leraning algorithm

In our neighborhood complete graph, we deal with transition probability. It is not necessary to have an initial model of

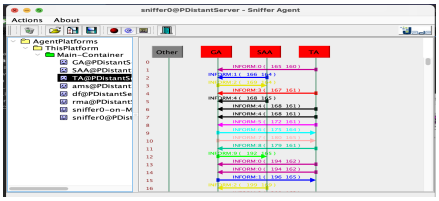


Fig. 5. Jade Sniffer Tool: Communication between GAA, SAA and TAA

the environment, we can then use $Q-Learning$ which is a model free learning algorithm to cope with the uncertainty of patients' arrival flows at any time in the ED. Our goal is to find an optimal policy for the patient scheduling problem in order to minimize patient waiting time. This $Q-Learning$ algorithm allows us to follow a policy which indicates what action a to take in which state s of the system. This works by a learning a state-action value function, denoted Q which defines the potential gain i.e., the long turn reward brought by the fact of carrying out a certain action a in a certain state s by following an optimal policy π.

$$Q_\pi(s,a) = E_\pi\{\sum_{i=0}^{\infty}\{\gamma^t r_{t+k+1} \backslash s_t = s, a_t = a\}\} \qquad (3)$$

In (3) , E_π is the reward expected by following the policy π. When this value state action is known by the agent, the optimal policy can be constructed by selecting the action with the maximum value for each state, i.e., by selecting the action a that maximizes the valu $Q(s,a)$ when the agent is in the state s. The agent follows any policy in order to update its value function. This algorithm allows, for any finite Markov Decision Process (MDP) to find an optimum policy of actions. The goal is, at each step of an episode, to maximize the value of the function $Q(s,a)$ defined as :

$$Q'(s,a) = Q(s,a) + \alpha(r + \gamma \max_{a\prime}(s',a') - Q(s,a)) \qquad (4)$$

The learning factor α represents the learning rate and determines how much the new calculated value function Q will outperform the old one. If $\alpha = 0$ the agent learns nothing, and if $\alpha = 1$ the agent considers only the last information he learnt. ($0 < \alpha < 1$). The discount factor γ determines the importance of future rewards. The factor of 0 makes the agent myopic by considering only current rewards while a factor close to 1 would involve more future rewards. If the discount factor is close to 2 or equal to 1, the value of Q may diverge.

As shown in the algorithm 3, the $Q-Table$ will converge to optimal $Q-Function$. In fact, the algorithm will plan the next action for the next state according to $Q(s,a)$ during iterations and update the new $Q(s',a')$. The iteration of the $Q-Learning$ will maximize the reward and produce the optimal policy.

V. COMPUTATIONAL EXPERIMENTS

In this section we present the computational experiments performed in order to evaluate and test the ABOS framework. For

Algorithm 2: Algorithm Regulate_SMS_Diversity (CS, NB, R, DT)

Input
CS: current solution to be inserted in SMS
NB: number of solutions in SMS
R: minimum number of different slots (containing different care tasks) for one solution to be considered different to the other
DT: diversity threshold
Max_{SMS}: maximum number of solution can be included in SMS
Output
possible insertion of CS in SMS
Begin
$d \leftarrow 0$ // number of solution in SMS different to CS
for $k = 1$ *to* NB **do**
  // browse all SMS solutions;
  $\lambda\_slot = 0$ // number of slots containing different care tasks in the SMS solutions compared to CS ;
  for $t = 1$ *to* T **do**
    for $j = 1$ *to* J **do**
      for $i = 1$ *to* I **do**
        if $(|CS^t_{i,j} - S^t_{k_{i,j}}| = 1)$ **then**
          $\lambda\_slot = \lambda\_slot + 1$
        end
      end
    end
  end
  if $(\lambda\_slot \geq R)$ **then**
    $d = d + 1$
  end
end
if $(\lambda\_slot = 0)$ **then**
  The solution is already exist in the SMS;
else
  if $(d/NB \geq DT)$ **then**
    if *(fitness(CS) is better then the fitness of the worst solution in SMS)* **then**
      if $(NB < Max_{SMS})$ **then**
        insertion the CS in the SMS;
      else
        eliminate the worst solution in SMS ;
        insert CS in SMS;
      end
    end
  end
end
end

Algorithm 3: Self Adaptive Agent (x_0) // based on Q-Learning Algorithm

Input
x_0: current solution
α: is the rate of learning
γ: is the discount factor
Output
x : learning solution
begin
Use $(Q(s,a)$, $\forall s, a)$
$\forall a$, $(Q(S_{terminal}, a) \leftarrow 0$
$x_{min} \leftarrow x_0$; $x \leftarrow x_0$
1 **while** *(Solution not improved)* **do**
  ▷ Start Episode;
  initialize the state s;
  $r = 0$;
2 **while** *(s is not a terminal state)* **do**
    if $r > 0$ **then**
      $s' \leftarrow$ SelectAction $(s, True)$ (algorithm 4);
      // an action a' from s by using the specific policy by Q (example : ϵ-greedy)
    else
      $s' \leftarrow$ SelectAction $(s, False)$ (algorithm 4);
      // no improvement: exploration other solutions
    end
    $x \leftarrow$ bestNeighbour $(s'$, $x)$
    if *x is better than* x_{min} **then**
      $r \leftarrow ((x_{min}$.getwaitingTime)-(x. getwaitingTime));
      $x_{min} \leftarrow$ x ;
    end
    $Q(s', a') = Q(s, a) + \alpha(r + \gamma \max_{a\prime}(s'a') - Q(s, a))$;
    $s \leftarrow s'$;
    $a \leftarrow a'$;
  end
end
return x
end

Algorithm 4: SelectAction (e, b)

Input
e: current state
b: boolean parameter
Output
$next\_e$: next state
Begin
if $(b = true)$ **then**
  $next\_e \leftarrow MaxAction(e)$; // ϵ_greedy exploitation
else
  $next\_e \leftarrow randomAction(e)$; // exploration
end
return $next\_e$
end

TABLE I
COMPARISON BETWEEN WAITING TIME USING GA, SAA AND TA WITH LEARNING AND WITHOUT LEARNING AND THE PRACTICAL CASE

Days	Number of patients		Genetic algorithm		simulated Annealing Algorithm		Tabeau Algorithm		Practical $\overline{W}$ (min)
	Scheduled patients	Unscheduled patients	$\overline{W}$ without learning (min)	$\overline{W}$ with learning (min)	$\overline{W}$ without learning (min)	$\overline{W}$ with learning (min)	$\overline{W}$ without learning (min)	$\overline{W}$ with learning (min)	
1	8	48	205.3	108.2	242.2	242	218.4	215.9	245.72
2	17	50	212.6	171.3	216.4	173.5	216	213.5	218.47
3	20	44	230.5	228.5	267.8	195	241.9	239.4	266.39
4	28	31	375	363.7	379.8	367.2	378.1	364.5	359.08
5	6	58	197.5	180.5	215.2	197.9	205.6	203.1	215.65
6	12	105	209.2	188.3	303.4	201.6	300.4	297.9	285.05
7	12	70	222.6	222.9	230.2	208	229	226.5	231.04
8	14	38	223.5	208.5	265.4	229	249	246.5	270.18
9	10	29	278.4	276.4	296.2	215.7	289.6	287.1	305.26
10	18	24	187.6	183.2	198.5	182.5	193.5	191	209.63

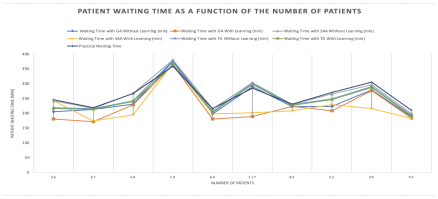


Fig. 6. The waiting time as a function of the number of patients

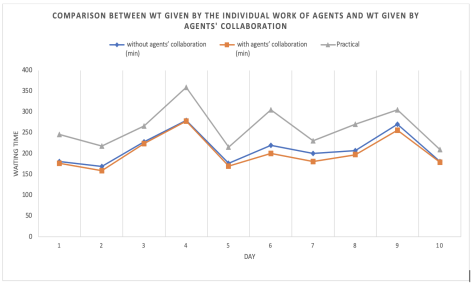


Fig. 7. The waiting time given by agents' collaboration and individual work

the development, we use the JADE (Java Agent Development Framework) platform based on java oriented-object language. This platform provides several packages practical for the development of MAS and provides graphical tools such as the sniffer agent to debug messages exchange between the agents (Fig.5). We use a real database of the AED provided by the LUHC. Data are collected thanks to ResUrgence, a software implemented in LUHC. We analyzed our computational results through the investigation of the effectiveness of the proposed self-adaptive system and its validation. Then, we evaluate its

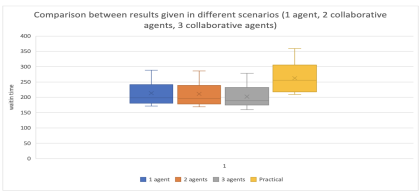


Fig. 8. Waiting time given in different scenarios

performance. For tests, we applied our approach to solve 10 problem instances, generated randomly, with different numbers of patients. These instances were generated on the basis of real data provided by the medical staff of LUHC. All patients coming to the AED should be treated in the current scheduling horizon or the next scheduling horizon H. In the present work, we assume that the duration of the scheduling horizon is 4 hours. The different tests are carried out using the same test conditions. In these tests, our Self Adaptive Algorithm is called in order to perform the local search. For the experiments, in the Q-Learning, we choose $\gamma = 0.87$, $\alpha = 0.13$ and $\epsilon = 0.07$. The objective of these simulations is to analyze the proposed algorithm performance, and thus to evaluate if the learning approach, which is integrated into the agent behavior, impacts the performance of ABOS framework in regards to the quality of the final output from an individual and a collaborative perspective. In order to evaluate the level of performance of our approach, we start to evaluate the individual learning added to the behavior of each agent in order to check if it improves the performance of our ABOS framework, that is, regarding individual learning, we report the results given when a single agent uses the Q-Learning algorithm. Then, we analyse the quality of the solutions obtained and we compare the results given by each agent that explores the advantages of a metaheuristic individually without using the Q-Learning algorithm to the results given by the individual learning. We did a comparison of the results obtained in practice (according to the ED database used by the medical staff) with those generated by the 3 metaheuristics (GA, SAA, TA). Table I shows the real AED data related to the test problem scheduling, together with the results obtained with the 3 metaheurisctics based approach, and the practical case. The gap between the solutions (related to patients' mean total waiting time per instance corresponding to several horizons H per day is shown in (Fig.6). Table I shows that the use of metaheuristics is associated with the minimization of the total mean waiting time, especially when using the learning algorithm. Fig.6 shows that the values of Q-Learning solutions are significantly better than solutions found by using metaheuristics only. For the different instances, the average waiting time of patients has markedly decreased thanks to the

use of metaheuristics and our proposed adaptive algorithm. The gap between the solutions is around 15%. However for the 2 instances 4 and 6, the mean practical waiting time is better than the mean waiting time given by the simulations because of the interferences between scheduled and unscheduled patients (particularly those requiring urgent treatment) arriving at the ED, which prompt real-time rescheduling. This is mainly due to the current mode of operation in the AED. In fact, the medical staff interrupts the ongoing care of a patient to treat a more serious unscheduled patient and he does not become available again until the treatment of this unscheduled patient is completed. The performance of the individual learning and the collaborative learning of the agents in the presented multi-agent environment were also assessed. Table I presents this analysis. We analyze the influence of learning in the cooperative process (Fig.7). Simulations show that Self Adaptive Algorithm when metaheuristic agents collaborate is better than the individual learning (without interaction). Regarding the use of two or 3 agents in collaboration to solve the problem, simulations show that there is a clear evidence that Self Adaptive Algorithm is better than using metaheuristics only. We also compared the given mean waiting time of patients in the AED of LUHC regarding the scenario in which an agent works individually exploring a metaheuristic (GA), with the solutions given by the two other scenarios of collaboration(2 agents, then 3 agents). The figure 8 shows that in collaboration, the more agents we have, the better the solution is. That is, the values of the total mean waiting time given by Self Adaptive Algorithm are better than those found individually, whether considering a single agent or considering two or more agents. In addition, the values get better when the number of agents involved in the search for the solution increases.

VI. CONCLUSIONS AND FUTURE WORKS

This paper presented a framework ABOS based on a collaborative multi-agent system (MAS) for multi-skill health care scheduling in the AED of LUHC using metaheuristics. 3 metaheuristics (GA, SAA, TA) are embedded in agents collaborating together in order to explore the research space and find out the optimal solutions. Each agent explores autonomously the advantages of a metaheuristic and interact with the other agents for the solution improvement. We also proposed in this article a self adaptive approach based on a learning process in order to allow the agents to adapt their actions according to their environment, their experiences and their interactions with each other. We adopted a Q-Learning algorithm. Simulations show that our approach has improved the AED performance by reducing the waiting time of patients. The results obtained show that the collaborative Learning approach leads to better results compared to the scenario in which agents work individually or without learning. In our future work, we will extend the neighborhood structures which will produce augmented states. Because it is crucial to accurately provide a correct decision for patients care in emergencies, we will explore the advantages of Deep Reinforcement Learning for improving the robustness and performance of ABOS framework. It will also be possible to study the impact of different learning approaches embedded simultaneously in agents' behaviors.

ACKNOWLEDGMENT

This work is part of Inter and Intra Hospital Logistics (OI-ILH) project supported and financed by the National Research Agency and is performed in the AED of LUHC (Lille, France). Also we would like to thank the domain experts in RHC of Lille for helping with field investigation and data collection (https://anr.fr/Project-ANR-18-CE19-0019).

REFERENCES

[1] Othman, S. B., Ajmi, F., Zgaya, H., & Hammadi, S. (2019). A cubic chromosome representation for patient scheduling in the Emergency Department. RAIRO-Operations Research, 53(5), 1453-1474.

[2] González, J., Ferrer, J. C., Cataldo, A., & Rojas, L. (2019). A proactive transfer policy for critical patient flow management. Health care management science, 22(2), 287-303.

[3] Ajmi, F., Othman, S. B., Biau, H. Z., & Hammadi, S. (2018, August). Scheduling Approch to Control the Execution of the Patient Pathway Workflow in the Emergency Department. In The 3th International FLINS Conference on Data Science and Knowledge Engineering for Sensing Decision Support (FLINS 2018).

[4] Rohokale, V. M., Prasad, N. R., & Prasad, R. (2011, February). A cooperative Internet of Things (IoT) for rural healthcare monitoring and control. In 2011 2nd international conference on wireless communication, vehicular technology, information theory and aerospace & electronic systems technology (Wireless VITAE) (pp. 1-6). IEEE.

[5] Ajmi, F., Zgaya, H., Othman, S. B., & Hammadi, S. (2020, October). Generic agent-based optimization framework to solve combinatorial problems. In 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC) (pp. 950-956). IEEE.

[6] Ajmi, F., Ajmi, F., Othman, S., Zgaya, H., Renard, J. M., Smith, G., & Hammadi, S. (2021, October). Friends and enemies agents collaboration protocol to optimize multi-skills patient scheduling in emergency department. In IEEE Systems, Man, and Cybernetics Society.

[7] Lohi, S., & Tiwari, N. (2021). Preliminary study of embedding two-level reinforcement learning to enhance the functionality of setting objectives compared with machine learning. Materials Today: Proceedings.

[8] Amiri, M. M., & Gündüz, D. (2019). Computation scheduling for distributed machine learning with straggling workers. IEEE Transactions on Signal Processing, 67(24), 6270-6284.

[9] Gaafar, M., Shaghaghi, M., Adve, R. S., & Ding, Z. (2019, November). Reinforcement Learning for Cognitive Radar Task Scheduling. In 2019 53rd Asilomar Conference on Signals, Systems, and Computers (pp. 1653-1657). IEEE.

[10] Benda, F., Braune, R., Doerner, K. F., & Hartl, R. F. (2019). A machine learning approach for flow shop scheduling problems with alternative resources, sequence-dependent setup times, and blocking. OR Spectrum, 41(4), 871-893.

[11] Hu, Z., Li, B., & Luo, J. (2017). Time-and cost-efficient task scheduling across geo-distributed data centers. IEEE Transactions on Parallel and Distributed Systems, 29(3), 705-718.

[12] Samma, H., Lim, C. P., & Saleh, J. M. (2016). A new reinforcement learning-based memetic particle swarm optimizer. Applied Soft Computing, 43, 276-297.

[13] Silva, M. A. L., de Souza, S. R., Souza, M. J. F., & Bazzan, A. L. C. (2019). A reinforcement learning-based multi-agent framework applied for solving routing and scheduling problems. Expert Systems with Applications, 131, 148-171.