International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 6 (2014) pp. 615-625

A novel approach to resolve the Dynamic Discrete Berth Allocation Problem in a Container Terminal

Sami Mnasri[1] and Nejah Nasri[2]

[1] Université de Toulouse, CNRS-IRIT Laboratory, IRT group,
IUT Blagnac-1 Place Georges Brassens, 31700 Blagnac-Toulouse, France
Sami.Mnasri@fsgf.rnu.tn

[2] Ecole Nationale des Ingénieurs de Sfax, LETI Laboratory, ISCSF group,
Sfax 3038, Tunisia,
nejah.nasri@isecs.rnu.tn

***Abstract*: In this paper, we are interested in the modeling and the resolution of the dynamic and discrete berth allocation problem which is noted DDBAP. To resolve this problem, we propose a heuristic approach of optimization which combines two concepts: agent and heuristics. This approach is based on the use of the multi-agent negotiation, the contract net protocol, and a set of heuristics such as the WorstFit arrangement technique and the LPT policy. The objective of our work is then, to solve the problem of scheduling n tasks on m parallel identical machines. The criterion that we aim to minimize is the makespan (in analogy with the P||Cmax problem) having a set of constraints to be satisfied. We developed our model of negotiation using the Jade platform. We finish this work by presenting various simulations to show the performance of the proposed heuristic and the contribution of our approach compared to other already existing approaches.**

***Keywords*:** Scheduling, Parallel machines, DDBAP, P||Cmax, MAS, Negotiation.

I. Introduction

A. Context and problematic

The transport problems are usually with a complexity that makes their resolution difficult and require the use of approximate methods and heuristics to resolve them. In this context, this paper is interesting in two main themes: scheduling and maritime transport. Given its economic and environmental objectives, companies have devoted significant efforts to the scientific research in scheduling, in order to upgrade and improve systems that aim to allocate and schedule tasks to resources. However, the resolution of the scheduling problems becomes quickly difficult considering the space, temporal and economic constraints taken into account.

Nowadays, the maritime transport becomes one of the fundamental pillars of the global economy. In general, the taken decisions in a maritime port are related to planning and control and are time-dependent. We distinguish three decision levels: strategic, tactical and operational level.

- The strategic level: it concerns the long-term decisions and focus on the design and the terminal infrastructure. These decisions are planned for several years.
- The tactical level: it is related to the decisions in means and short term (few days to few months) and concerns the organization of the terminal operations such as resources allocation, human resources management, allocation policy and internal transport.
- The operational level: it is related to the instantaneous decisions of each day aiming to achieve the various tactical level tasks [1].

The berth allocation problem is attached to several decision levels, especially the tactical and operational ones.

The complexity of management of a container terminal and the mentioned decisions levels, introduce several optimization problems where each problem is focused on one or more operation in the terminal. These problems are classified among the transport and scheduling problems, such as the vehicle routing problem and the assignment problems which are largely studied in the literature. We can classify these problems in four main classes:

* Ships planning:
  - Berth Allocation Problem (BAP).
  - Quay Crane Allocation Problem (QCAP).
  - Container Stowage Problem (CSP).
* Transport on the quay:
  - Course cranes scheduling.
* Management of the handling equipment:
  - Scheduling of the handling equipment.
  - Maintenance of the handling equipment.
* Yard Management:
  - Container Stacking Problem.
  - Empty containers repositioning.

Our works aim to resolve the problem of allocating berths to ships. The BAP is regarded as a parallel machines scheduling problem. A task and a machine can be considered as a ship and a berth, respectively. To resolve this problem, we adopted a heuristic approach based on the multi-agents negotiation, the contract net protocol, the WorstFit technique of arrangement and LPT policy. Indeed, the agent approaches are recently emerged as powerful technologies that are able to contribute in

the design and the development of complex systems. The capability of these approaches to solve complex problems is based on the agent's capacities, their autonomy and their adaptability to complete common goals by interacting together.

B. Objectives

The objective of our work is to study the scheduling problem on identical parallel machines. We aim to schedule n jobs on m identical parallel machines. The criterion we want to minimize is the makespan (Cmax).

C. Organization of paper

This paper is composed of five parts: The first part presents a global view on the studied problem. We introduce this problem by presenting the context, the aimed objectives and the various levels and decision problems involved in the container terminal operations. In the second part, we present the dynamic and discrete berth allocation problem and its state of the art. We present also scheduling problems, its resolution methods, the P||Cmax problem and its state of the art. The third part presents the suggested solution. We present the resolution approach and its description. The fourth part presents the experimentation, the development environment and the obtained numerical results. Finally, we present a conclusion which summarizes our contribution, highlights some observations and prospects to improve our results.

II. State of the Art

A. State of the art on the BAP

A berth is a location, usually with a well defined length in the quay where a ship can land to perform handling operations (loading / unloading containers). Berths are generally equipped with specific installations to facilitate the containers handling. We note that berths are at the heart of the purpose of our work, we are interested in optimizing the allocation of berths to a set of waiting ships in the port knowing that berths represents a key factor to have an efficient port service. This problem consists in assigning ships to quay sections in the container terminal, respecting a planning time. Indeed, as indicated in figure 1, inspired from Cordeau and Laporte works [4], a set of ships arrive to the container terminal in different but known in advance dates. It is necessary to determine precise schedules for the entry of these ships (a scheduling problem) and a berth (an assignment problem) where handling operations will take place. Several restrictions and constraints must be taken into account such as quay length and containers loading / unloading time. Due to these restrictions, berths become critical resources that imply waiting times to have access to them. However, profitability and productivity of the port requires the reduction of the waiting time and berthing costs. This very practical problem has been the subject of several studies in the operational research according to various approaches.

Indeed, the berths to ships allowance is the first planning operation in the container terminal, it is a major factor in measuring performance and efficiency of a terminal. Researches related to this problem are justified primarily by the ships accosting time passed in the quay. This time is considered by the ship-owners as a lost profit for each hour in late passed in the quay, since this time is a period without generation of revenues [3]. The principal decision to take during accosting is the choice of "where" and "when" ships must accost. Thus, this planning determines position and time for each vessel arrival in order to minimize the total accosting time, knowing the estimated arrival time and the operations duration which represent the temporal aspect of the problem.

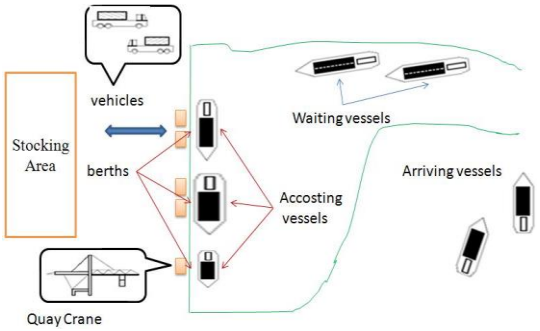

Figure 1. Ships accosting

This problem can be modeled in different ways: according to the vessel arrival and the service time (a static or a dynamic problem) or according to the accosting space (a discrete or a continuous problem). This is due to the various assumptions and constraints involved in the modeling of the problem. This problem is known as dynamic (DBAP) if the vessel assignment process starts whereas some ships did not arrive yet at the port but we know their arrival times. The problem is considered static (SBAP) if all ships to serve are in the port before the assignment beginning and before berths becomes available [4]. It is known as a discrete problem if the quay is represented as a finite set of berths where each berth is located by points or segments having the same length and each berth can serve one ship at a given time. It is known as a continuous problem if we assume a variable ships length, a variable cranes capacity or when we authorize the ships accosting anywhere along the quay. In our works, we focus on a significant variant of this problem which is the dynamic and discrete one, noted DDBAP.

Several studies and researches treat the dynamic and discrete case problem. Among others we mention those of Imai and al. [5] in 2001, where they introduce the static and the dynamic problem by proposing a heuristic approach based on a lagrangian relaxation and aiming the minimization of the total service time which is composed of a waiting time and a handling one. In the same year, Nishimura and al. [6] proposed a genetic Algorithm based on a set of heuristics to solve this problem having the same objectives and including a set of physical accosting restrictions such as the water depth. Also, Guan and Cheung [7] presented in 2004 a set of heuristics to minimize the total weighted flow time by considering that flow time is composed of a waiting and a processing time whereas the weighted time reflects the importance of ships. Cordeau and al. [8] introduced an algorithm based on the tabu search strategy to solve this problem, but only the small sizes of the problem were solved. Recently, in 2008, Imai et al. [9] proposed a genetic algorithm based on heuristics to minimize the total service time. In the same year, Hansen and al. [10] introduced a new allocation policy to resolve the DDBAP, which is based on a variable neighborhood search heuristic. In 2009, Golias et al. [11] formulate the DDBAP as a multi-objective combinatorial

optimization problem and they developed a genetic algorithm to solve it. Also, Theofanis and al. [12] solve the same multi-objective combinatorial optimization problem by using an alternative of the genetic algorithm proposed by Golias et al. In 2012, Mnasri S. and Zidi K. [13] proposed a negotiation model based on agents to solve the DDBAP, our works are the extension of this paper.

This problem is related in several other decision problems in the container terminal. Mainly the quay crane allocation problem (QCAP) [14], the simultaneous optimization of the berth assignment and the quay cranes problem (QBAP) [15] and the optimization of the container to AIVS assignment problem [16].

B. State of the art on the scuduling

Scheduling is a scientific discipline studied since fifties which consists in organizing the execution of a set of tasks, taking into account temporal constraints (time, sequencing, etc.) and other constraints related to the use and the availability of required resources. Thus, scheduling concerns the allocation in time, of a set of limited resources to tasks. It is a decision-making process aiming to optimize one or more objectives. These objectives varies according to the treated scheduling problem and can be related to the minimization of the total tasks execution time (the makespan), the minimization of the number of tasks to be carried out after their due date, or the minimization of waiting times, etc.

The scheduling problems belong to the under constraints combinative optimization problems. The majority of the scheduling problems are NP-Hard. Our problem is classified NP-Hard considering the complexity of the accosting constraints which include several factors and parameters. Even if it is about only one berth, the problem is proved NP-Hard by Lenstra and al. [17].

We model our problem (DDBAP) as a problem of scheduling tasks on identical parallel machines in order to minimize the makespan. In the scheduling literature, this problem is known as P||Cmax. Indeed, the parallel machines are classified according to their speed. They are qualified as identical if all machines have the same execution speed.

In 1979, Graham [18] proposes the $\alpha / \beta / \gamma$ notation which becomes a reference in characterizing the scheduling problems: The α field describes the structure of the problem and it is divided into two sub fields $\alpha 1$ and $\alpha 2$, $\alpha 1$ references the problem nature (Flow Shop,..), and $\alpha 2$ specifies the machines number. The β field describes the types of constraints taken into account. The γ field presents the considered objective function. Generally, criteria to be optimized depend on the products completion dates. To describe the DDBAP, we consider the following values:

- Field α1 Value: P: indicates that machines are **parallel** and **identical.**
- Field β Values:
  - *ri* A beginning earliest date ri is associated to each task t*i.*
  - *di* A preferred due expiration date *di* is associated to each task t*i.*
  - *No wait* *The* operations of each specific task must be executed without waiting.
  - *Snsd (Rsnd)* Resources must be prepared before and/or after each task execution, independently of the tasks sequence.
- Field γ value: the *Cmax* which references the *makespan*.

Then, our problem will be noted as follow: **P| ri di No-wait Snsd | Cmax**.

The resolution methods of the scheduling problems are related to several scientific disciplines. Among them, we find the operational research where many resolution methods are based on it. We find exact methods like dynamic programming and the branch-and-bound. The constructive heuristics and greedy algorithms are also part of it, among them we find list algorithms and priorities rules. Constructive heuristics are iterative methods which generate solutions by the addition of elements in each iteration. Also, there are other methods derived from operational research like the local search. The main idea of these methods is to start from a realizable solution, and then gradually modify it until a stop criterion is satisfied. The subtlety of these methods is based on the generation of a close to optimal solution, as well as its acceptance rules. As example of the local search methods we mention the descent method, the tabu search and the simulated annealing. We can also develop metaheuristic methods to solve scheduling problems. Heuristics and metaheuristics are considered as approximate methods [19]. Recently, multi-agents systems and distributed artificial intelligence proved their efficiency in the resolution of the scheduling problems.

In the literature, P||Cmax is treated by several research works, we motioned the following works: In 1996, Fran ça and al. [20] used a tabu search strategy to solve it. In 2003, Antonio Frangioni and Emiliano Necciari [21] proposed approximations and heuristics based on the multi-exchange neighborhood for the P||Cmax. Also, DellAmico and Martello [22] in 2005 proposed an exact method based on a Branch and Bound algorithm. In 2008, Iori and Martello [23] solved this problem using a scatter search algorithms. In 2010, Adriana c.f. Alvim [24] proposed hybrid heuristics combined with a tabu search strategy. In what follows, we compare our results with those of DellAmico and Martello and those of Adriana C.F. Alvim.

III. The Suggested Solution

A. The resolution approach

To solve the DDBAP, we use a distributed approach combining two concepts: Agent and Heuristics. Indeed, multi-agent systems constitute a non recent research field. However, they represent a very active research field. Indeed, it is a distributed system composed of a set of intelligent agents which are autonomous, able to interact and to be organized. Thus, these systems give agents the possibility to cooperate and to coordinate their goals and action plans in order to satisfy and solve a specific problem. The modeling of the based on agents systems consists on the establishment of a multi agent organization to satisfy aimed objectives. Indeed, it is about classifying and combining various tasks and equipping the agents with a set of competences (roles and knowledge) enabling them to interact according to a communication protocol (the Contact Net for example) in order to achieve the organization common objective. It is also about describing complex agent behaviors which govern its

relations with the other agents and the environment. Recently, the development of the distributed control theory allowed several researches to apply the multi-agents system theory to solve the container terminal problems. Indeed, a multi-agent system can reflect in a realistic and generally effective way, the complex relations between the system actors and the way in which they are organized. The multi-agent system can also provide a solution which is acceptable by the majority of implied actors in the system. This can be very useful since, in many practical situations, the problem data are not completely known from the beginning.

The problem of the scalability is also less constraining when we use the agent approach. We note that the system environment is complex, highly dynamic and decomposed into subtasks. This imply taking into account the distributed structure as well as the data flow circulating between the various actors of the system (ships, berths, Etc) which are largely influencing the ships scheduling and the satisfaction of all requests in a dynamic context. Also, the multi-agent approach gives the possibility to explore a dimension search space which is, in most cases, not searchable via the traditional resolution methods. Thus, we obtain a global solution which is incrementally built starting from the partial solutions provided by the agents. Indeed, each agent searches a locally feasible solution and negotiates its neighbors in order to make it coherent.

The negotiation is a mechanism of coordination and interaction between agents which allows improving the agreements about the points of view and the action plans after the exchange of interesting information [25]. This mechanism includes a protocol to organize negotiations, a communication language and a decision-making process which allows to each agent to decide his position, his choices and his orientations. The negotiation concerns the distributed resolution of the conflicts and the decision-making. After the negotiation, an offer is accepted, refined, criticized, or refused. The negotiation comprises several models. These models are either theoretical such as the vote and the game theory models, or data-processing models such as the knowledge based negotiation, the negotiation led by constraints or the Contract Net of FIPA. The Contract Net is one of the multi-agents approaches which aim to resolve the tasks allowance problems. This protocol establishes a contract between a supervisor called manager and a set of contractor agents. Indeed, as schematized in figure 2, the manager decomposes tasks into several subtasks and diffuses each subtask on all contractor agents. Contractors receive the task announcement and propose their offers to the manager. These offers reflect their capacities to realize the desired task. The manager gathers the proposals and allocates the task to the agent which gives the best proposal. Then they exchange informations until the task achievement. In extreme cases, there will be, the manager cancels the contract and the task execution.

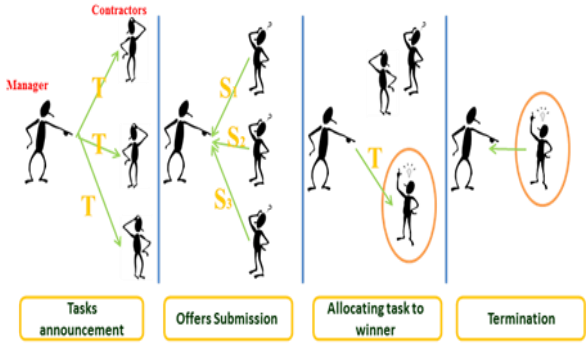

Figure 2. The contract net phases

1) Inter-ships negotiations

In addition to the negotiation via the contract net protocol, our system comprises a negotiation mechanism between the ships agents. Indeed, we aim to respect the constraint which implies the dynamicity of the berth allocation problem. This constraint supposes that scheduling can starts while there are some ships which are not yet present physically in the port but their arrival times are known in advance. Then, we must be sure that the ship planning (its handling date) will not precede its arrival date. In this case, an inter-ships negotiation mechanism starts if a ship handling date precedes its arrival date. Indeed, the concerned ship tries to delay its handling date to the profit of the next ship in the planning without violating its preferred due date. This ship communicates with other ships to delay its handling date until respecting the previous indicated constraint.

2) Used Heuristics

To solve the DDBAP, we use two heuristics, the first one is LPT policy and the second is the WorstFit arrangement technique.

The longest processing time (LPT) is a known policy which allocates tasks to machines in a descending order of their processing times. This algorithm is the most current used heuristic in the literature for the parallel machines planning and the makespan minimization problem which is our objective. Indeed, the interest of the tasks is measured by the number of containers to handle. We assign to each vessel, a berth according to the number of containers to handle, more worked containers, more incomes. In our case, the ships execution times are supposed to be known in advance, because ships call the terminal before landing in order to reserve a berth.

Several heuristics was proposed to quickly find possible solutions to the problem of arrangement of a set of objects having different widths in a set of boxes. As an example, we mention the *First-FIT* (FF) heuristic which sequentially places objects on the first appropriate box. The *Best-FIT* (BF) which also, sequentially traverses objects but it place them in the box which has the smallest sufficient available capacity and the *Worst-FIT* (WF) which always places objects sequentially in the box having the greatest sufficient available capacity. Indeed, the *Worst-FIT* technique is used to resolve the problem of arranging objects and to inserting tasks in empty locations in order to minimize their makespan. This heuristic is also used to solve another traditional problem which is the *Bin-Packing Problem*. This problem consists in arranging objects in the minimum possible number of identical boxes. These two problems are very dependent. Basically, they

correspond to the same decision problem but they have different objectives to minimize. Figure 3 illustrates the passage from an arrangement to a scheduling according to the Worst-Fit technique.

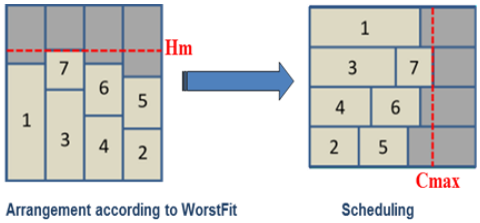

Figure 3. Transforming an arrangement to a scheduling

Among the used techniques to represent a scheduling, we have the Gantt diagram which constitutes the simplest mean and the more used one to represent a scheduling. Indeed, this diagram enables us to represent the affectation of tasks (ships) to machines (berths) and to show the machines time occupation, the treatment sequences on each machine and the completion dates. The figure 4 presents an accosting plan in time and space.

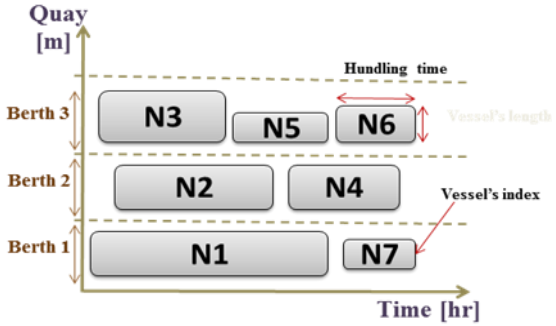

Figure 4. A Space-time representation of an accosting

B. Detailed description of our approach

After presenting the problem, we will deduce its modeling.

1) Notations and decisions variables

We describe the scheduling problem using the following notation and decision variables characterizing the system:

I the number of machines.
T the number of tasks.
i (= 1...I) ∈ B the set of berths (set of machines).
j (= 1...T) ∈ V the set of ships (set of tasks).
k (= 1..T) ∈ O the set of the service order.
$\mathbf{S_i}$ time when berth I becomes empty and can be included in the planning.
$\mathbf{A_j}$ arrival time of the ship j.
$\mathbf{C_{ij}}$ handling time passed by the ship j in the berth i.
$\mathbf{X_{ijk}}$ 1 if the ship j is served as the K^{th} ship in the berth i, 0 else.
$\mathbf{P_K}$ subset of O knowing that $P_K = \{p|p < K \in O\}$.
$\mathbf{W_i}$ subset of ships knowing that $A_j \geq S_i$.
$\mathbf{Y_{ijk}}$ free time (of the berth i) between the departure of the $(k-1)^{th}$ ship and the arrival of the k^{th} ship when the ship j is served as a k^{th} ship. :

2) Objective function

We aim to resolve the P| ri di no-wait Snsd | Cmax. Our objective to minimize is the makespan of the tasks (Cmax). As a result, we have the following objective function:

Minimizing

$$\sum_{i\in B}\sum_{j\in V}\sum_{k\in O}\{((T-k+1)Cij + Si - Aj\}Xijk + \sum_{i\in B}\sum_{j\in V}\sum_{k\in O}(T-k+1)Yijk \quad (1)$$

3) Constraints and restrictions taken into account

To specify our problem, we detail several constraints which express restrictions on the values that the decision variables can take. Among the considered constraints we have:

(a) Time of changing ships

It is often supposed in the literature that times needed to moor or to leave the berth are negligible. In our problem, we suppose a constant time between the times of service of two consecutive ships using the same berth.

(b) The non-preemption:

The preemption consists in dividing tasks into a set of subtasks to allow the parallel execution of independent tasks. We suppose that pre-emption is not allowed since the migration of ships and its displacement between berths generates additional mooring times which are often more important than the saved time. This means that once the ship is moored, it will remain until handling operations are finished.

(c) Initial and final cranes assignment

This constraint specifies which cranes are affected, their starting and final positions. Considering the nature of the treated problem (P||Cmax), we release this constraint and suppose that all berths have the same quay crane handling time.

Thus, the constraints of the problem (inspired from the dynamic formulation of Imai and Nishimura, [8]) can be formulated as follows:

$$\sum_{i\in B}\sum_{k\in O} Xijk = 1 \text{ for any } j \in V \quad (2)$$

$$\sum_{j\in V} Xijk \leq 1 \quad i \in B, k \in O \quad (3)$$

$$\sum_{i\in V}\sum_{k\in P_k}(Cik\ Xikm + Yikm) + Yijk - (Aj - Si)\ Xijk >= 0$$
$$\text{for any } i \in B,\ j \in Wi, k \in O \quad (4)$$

$$Yijk \geq 0 \text{ for any } i \in B,\ j \in V, k \in O \quad (5)$$

$$Xijk \in \{0,1\} \text{ for any } i \in B,\ j \in V, k \in O \quad (6)$$

The constraints (2), (3) and (6) concern the positioning of the ship. The constraint (4) ensures that serving ships should begin only after their arrivals. We suppose that C_{ij}, S_i and A_j have integer values. Also, Y_{ijk} may have an integer value that indicates the priority report between the ship arrival and its service. The constraint (5) indicates a free time of the berth i between the departure of the $(k-1)^{th}$ ship and the arrival of the k^{th} ship j. To explain that the service is carried out after the vessel arrival, Y_{ijk} is defined simply as the difference between the beginning of the service for the ship j and the departure of its immediate predecessor.

C. Detailed modeling of the proposed multi agent approach

1) Proposed multi agent architecture

We propose, in figure 5, the following architecture which comprises three types of agents in addition to the containers as objects: the mediator agent (scheduler), the berth agent, the vessel agent.

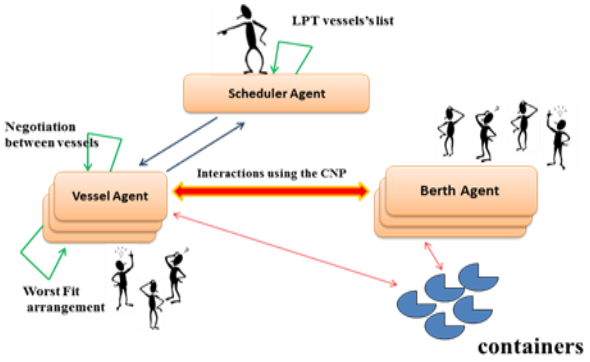

Figure 5. The proposed multi agent architecture

In what follows, we present the characteristics of each agent: its roles, knowledge and behaviors which are appropriated to their tasks.

(a) Roles and knowledge of the agents:

- ***Scheduler Agent:*** It is the agent which manages the negotiation process after the creation, starting from the graphical interface, of the berth and vessel agents. Thus, its role is to order ships in a queue according to a well-defined policy, to plan the berth allowance, the containers loading/unloading and to provide the scheduling results. It knows the total number of ships, the total number of berths, the identifier of each vessel agent and the identifier of the corresponding berth agent and its characteristics and the routing order of the ships in the built queue. It knows also the characteristics of each berth agent (its identifier, its length in meters, its depth, its availability, its current load, etc) and the characteristics of each ship (its identifier, length, depth, number of containers to be charged or discharged, estimated arrival date, preferred due expiration date, etc).

- ***Berth Agent:*** It is the agent which represents the quay location, its role is to receive accosting requests launched by ships; it is also able to calculate their offers corresponding to their capabilities to receive the concerned ship and to send these offers to ships. It knows its characteristics (identifier, length, depth, availability, current load in containers...) and characteristics of the ship agent asking for accosting (identifier, length, depth, number of containers to be charged or discharged, estimated arrival date, preferred due expiration date, etc).

- ***Ship Agent:*** It is the agent which represents the ship, its role is to search the suitable quay location to accost. It launches an accosting request to all berths and then it chooses the suitable berth. It knows the total number of berths, the identifier of each berth agent and its characteristics, the berth proposals following its requests, the identifier of the selected berth agent and its characteristics. It knows also its characteristics (identifier, length, depth, number of containers to be charged or discharged, estimated arrival date, preferred due expiration date, etc).

(b) Agent behaviors

Agents select machines on which these tasks should be treated. The way in which agents select tasks constitutes their behaviors. A behavior is an event handler which works as a method which describes how an agent reacts to an event. Formally, it is an appropriate change of state. In jade, behaviors are classes and the code of the event handler is placed in a method called **action.** Behaviors are either primitive or compound. Primitive behaviors are either simple, cyclic or with only one execution. Compound behaviors are either parallel, or sequential. The principal behaviors of our agents are:

- ***Scheduler agent behaviors:*** It creates new berth agents and vessel agents following the user action started from the interface, it initially orders vessels in a queue, gives them useful data to find a berth, displays results in a table, in a graphic area and in a text area containing the obtained scheduling details. It can also authenticate and provide information on itself.

- ***Berth agent* behaviors:** It receive the demand of each ship, evaluates this request using the behavior "evaluateAction()", prepare an offer "behavior: prepareResponse()", sends this offer, manages notifications "behavior: prepareResultsNotification()",manages the proposals acceptance or rejection "behaviors: hundleRejectProposal() and handleAllResponses()", receives the acceptance of the offer from the vessel, executes the requested task and sends a feedback indicating the termination of the task execution.

- ***Vessel agent* behaviors:** it sends requests to the agents using the behavior: "call for proposal (cfp)", manages the berth answers using the following behaviors: "hundleAllResponces(), hundlePropose(), hundleRefuse() and hundleFailure()". Also, it accepts or refuses each request "accept-proposal / reject-proposal" and it provides the offers results to the berth agents using the behavior: "hundleInform(): inform(done), failure or cancel".

2) OMASE Modeling

To model our multi-agent system, we use O-MaSE (Open Multi-Agent Systems Engineering): a methodology which follows the oriented object principle to design the multi-agent systems. Its purpose is to build an organizational society of agents based on the meta-model of the organization [26]. Each agent plays a specific role to achieve a goal according to its capacities. O-MaSE is the extension of MaSE. It takes into account the advantages of MaSE and those of the organization engineering. It also allows modeling a complex and opened multi-agent system. Also, because of its simple use and being recognized by various research works, we propose to follow this methodology and its associated platform (AgentTool) to conceive our System.

The development process of this methodology treats the following phases: analysis, design and implementation:

**** The phase of analysis:*** it consists in:

- Identifying goals: a goal is considered as the objective of the system. Goals are structured and organized according to their order of importance.
- Determining the desired system behaviors and describing the exchanged messages.
- Defining roles: each goal is transformed into a role. This role will be played by an agent.

**** The phase of design:*** it consists in:

- Creating agent classes: the agent classes are identified starting from roles.
- Building conversations: we define protocols between agents.
- Assembling agent classes: we create internal agent classes according to the selected architectural model (BDI or other).
- Defining the system structure: it is presented by the deployment diagrams of OMASE.

**** The phase of implementation:***

AgentTools facilitates the development of the multi-agents applications. It allows checking, generating code and reusing components. However, we do not limit ourselves to this automatic code generation, but we will implement our own system. This realization is one of our contributions and its results will be more detailed in the following section.

This methodology, as described in figure 6, contains several models like: Goals, Roles, Agents, Protocols and Plans. This figure presents also the development steps of these diagrams.

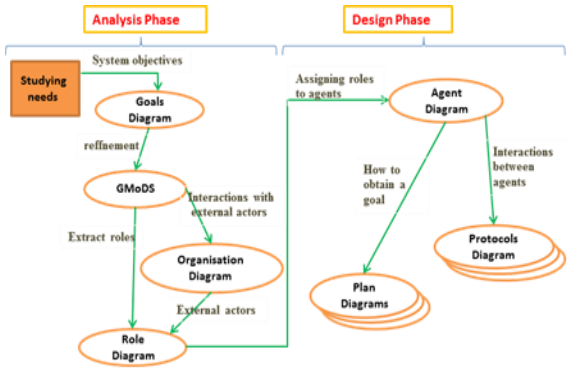

Figure 6. Phases of the OMASE methodology

IV. Experimentation

A. *Detailed Design and development Environment:*

To conceive our multi-agent system, we use AgentTool. To develop it, we use the Jade platform. AgentTool is a graphic environment which is developed in java to help developers to analyze, conceive and implement multi-agents systems. It is conceived to support the MASE methodology. At3, the last version of AgentTool, supports O-MaSE and provides the possibility to check the coherence between models and generating automatically the code according to the agent model. This environment, incorporated on the eclipse platform provides eight models. These models support analysis, design, and the execution of the multi-agent system according to the O-MaSE methodology. Also it provides the capacity to check and adapt processes to the required needs. Besides, aT3 provides a checking framework which helps designers to maintain coherence between the OMaSE models. Figure 7 illustrates the GUI of the agentTools III.

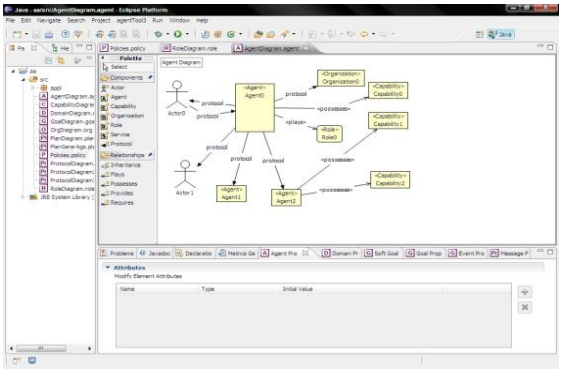

Figure 7. The AgentTools GUI

The *JADE* platform (*Java Agent DEvelopment Framework)* is an agent development environment fully implemented in the JAVA language. It facilitates the development of a multi-agent system that satisfies the specifications of *FIPA* (*Foundation for Intelligent Physical Agent)*. Jade provides a set of classes which define the behaviors of each agent behaviors. Agents use the FIPA ACL language to communicate. An editor is available to manage agents. Jade uses a set of services to ensure the conformity to the FIPA standards. Among these services we mention the following ones: the name service, the yellow pages service, the transported messages and the analyzing service; and the FIPA interactions protocol library. Communication is realized by a set of exchanged messages. FIPA ACL is used to represent messages. The agent platform can be distributed on many machines. On each machine, only a java virtual machine JVM is executed. Basically, each JVM is an agent container which provides an environment to execute agents. Figure 8 illustrates the Jade GUI.

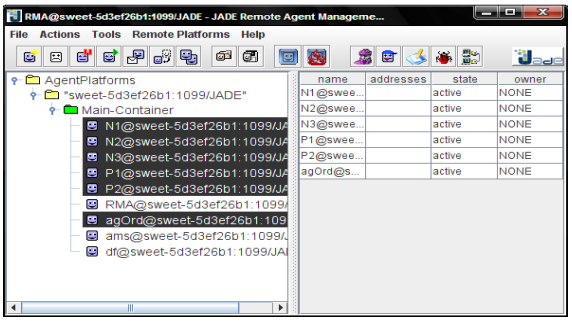

Figure 8. The Jade GUI

B. *Numerical results*

To validate our work, we compare our results to other works like those of Ethel Mokotoff, [28], and Mauro Dell' Amico and Martello in 1995 [27] and 2005 [22]. Along this comparison, we used a Pentium 600 MHz to be close to the environnment of Mookotoff and Dell' Amico. We treated the same problem as Ethel Mokotoff and Mauro Dell' Amico (**P||Cmax)** and we tested our work on the same platform and the same values but we take into account of a set of additional constraints (**P| r_i d_i No-wait Snsd | Cmax)**. This makes the problem more difficult to solve but does not increase its complexity.

1) Obtained results for the E.Mokotoff and DellAmico instances

(a) Obtained results for the small size instances of E.Mokotoff and DellAmico

In this section, we compare our results to those of E.Mokotof and Dell' Amico for the small size instances ($5 \leq n \leq 15$ and $3 \leq m \leq 5$); n is the number of tasks (vessels in our case) and m is the number of machines (berths). Indeed, to solve the P||Cmax problem, Dell' Amico and Martello [27] proposed an algorithm and its implementation in C, this algorithm is based on the branch and bound method with a criterion of dominance and a calculation of a lower and higher bound. We used three classes of random instances which uniformly generate values of execution times P_i (in seconds) of n tasks in the following three intervals [1,100], [10,100] and [50,100]. For E.Mokotoff [28], he tested his work (based on the branch and bound method) using a computer equipped by a Celeron 434 MHz with the same three classes of instances generated with the same way.

To test our system, we take an execution time which is the average of 10 execution times. To represent the beginning earliest date r_i, we use a parameter Q chosen in the set {1,5}, r_i is generated uniformly in the interval [0, Q*n], n is the vessel number. The preferred due expiration date di is generated automatically when creating each vessel agent. Table 1 compares our results compared to those of Mokotoff and Dell' Amico.

For small size instances, obtained values clearly demonstrate the advantage of our work compared to those of Mokotoff. However, they show the advantage the DellAmico and al. works on ours. The following histograms (figure 9, figure 10 and figure 11) show the obtained results starting from samples taken from table 1 representing the small size

Table 1. Obtained Results for the Small Size Instances

		$P_i \in [1,100]$			$P_i \in [10,100]$			$P_i \in [50,100]$		
M	**N**	**Mokotoff**	**Dell'Amico**	**Our System**	**Mokotoff**	**Dell'Amico**	**Our System**	**Mokotoff**	**Dell'Amico**	**Our System**
3	5	0.06	0.000005	0.000006	0.02	0.000007	0.000008	0.06	0.000010	0.0000012
3	6	0.01	0.000010	0.000011	0.02	0.000014	0.000008	0.06	0.000007	-
3	7	0.09	0.000024	0.000011	0.08	0.000038	0.000007	0.10	0.000019	-
3	8	0.10	0.000036	0.000008	0.09	0.000076	0.000008	0.12	0.000131	0.000009
3	9	0.05	0.000040	0.000009	0.12	0.000101	0.000009	0.11	0.000073	-
3	10	0.06	0.000100	-	0.12	0.000210	-	0.22	0.000278	-
3	11	0.14	0.000113	-	0.22	0.000355	-	0.54	0.000256	-
3	12	0.08	0.000157	0.000010	0.12	0.000112	0.000011	0.38	0.000361	-
3	13	0.14	0.000125	0.000011	0.19	0.000009	0.000012	1.38	0.000618	-
3	14	0.08	0.000012	-	0.15	0.000093	-	1.48	0.000590	-
3	15	0.22	0.000014	0.000015	0.19	0.000014	0.000015	0.27	0.000011	0.000017
4	5	0.01	0.00002	0.000021	0.01	0.000001	0.000028	0.01	0.000001	-
4	6	0.03	0.00003	0.000031	0.01	0.000003	0.000038	0.07	0.000008	-
4	7	0.02	0.000018	0.000033	0.02	0.000022	0.000044	0.05	0.000028	-
4	8	0.06	0.000021	0.000036	0.08	0.000017	0.000046	0.12	0.000021	0.000052
4	9	0.04	0.000076	0.000066	0.12	0.000102	0.000056	0.25	0.000047	-
4	10	0.03	0.000043	0.000076	0.30	0.000088	0.000089	0.74	0.0000373	-
4	11	0.09	0.000215	0.000282	0.90	0.000297	0.000282	0.99	0.001426	-
4	12	0.57	0.000210	0.000373	1.17	0.000406	0.000452	0.95	0.000448	0.000440
4	13	2.98	0.000537	0.000415	5.00	0.000841	0.000462	15.74	0.000921	-
4	14	0.62	0.000300	0.000484	0.14	0.000403	0.000468	39.84	0.000608	-
4	15	0.57	0.000290	0.000474	0.49	0.000256	0.000481	47.06	0.001546	
5	6	0.01	0.000002	0.000013	0.02	0.000001	0.000016	0.01	0.000001	0.0000178
5	7	0.01	0.000008	0.000016	0.01	0.000011	0.000015	0.03	0.000010	-
5	8	0.01	0.000013	-	0.02	0.000021	-	0.04	0.000028	-
5	9	0.02	0.000022	0.000023	0.08	0.000017	-	0.02	0.000030	0.0000342
5	10	0.05	0.000034	0.000036	0.06	0.000072	0.000032	0.06	0.000029	-
5	11	0.37	0.000092	0.000028	0.30	0.000133	0.000043	3.61	0.000231	-
5	12	0.09	0.000126	0.00022	1.81	0.000324	0.000363	17.96	0.000471	-
5	13	1.68	0.000183	0.00029	5.06	0.000359	0.000378	74.23	0.000930	0.00062
5	14	5.39	0.000201	0.00025	30.98	0.000414	0.00052	50.04	0.000926	0.00064
5	15	2.85	0.000247	0.00033	0.29	0.000925	0.00102	52.73	0.001007	-

instances of the three types of classes. We simply compare our results to those of Dell'Amico since, according to the table 1 and [22], Dell'Amico and al. results are better than those of Mokotoff for all instances. Missing entries, represented by (-), indicate that we have not tested these cases.

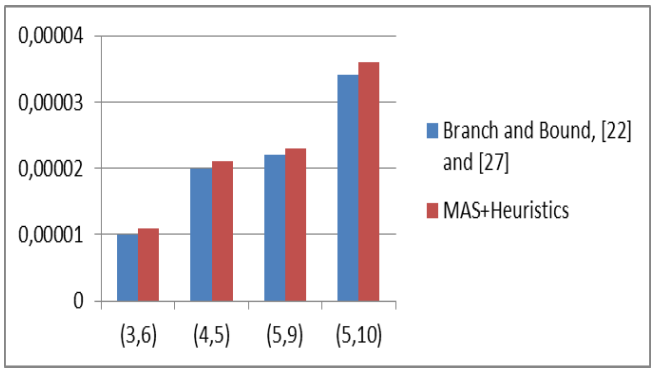

Figure 9. Small size instances results for the first class

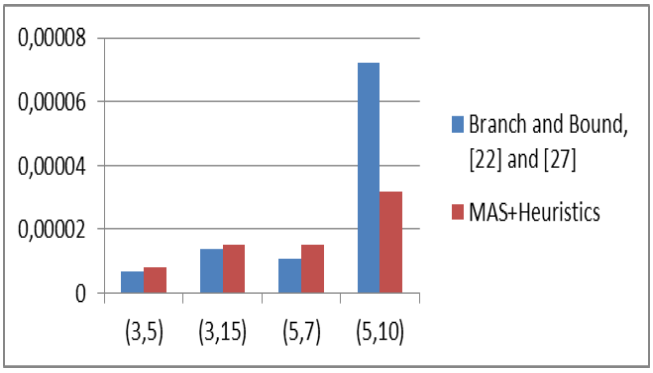

Figure 10. Small size instances results for the second class

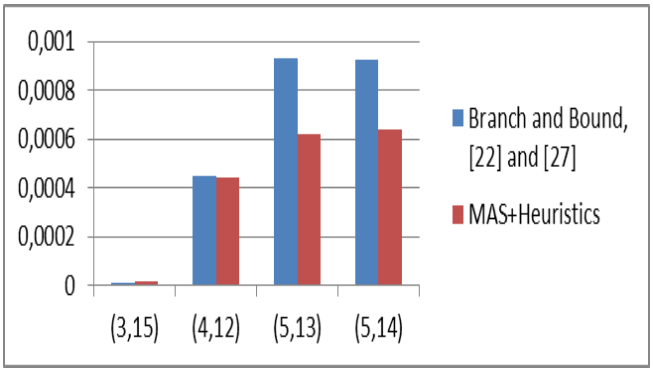

Figure 11. Small size instances results for the third class

(b) Obtained results for the big size instances of E.Mokotoff and DellAmico

Table 2 shows the results of same works using a larger size of instances ($20 \leq n \leq 1000$ and $3 \leq m \leq 100$). The execution time values of the tasks are taken in the interval [1,100].

The figure12 shows the obtained results for the big size instances for several values of m and n taken from the preceding table (table 2).

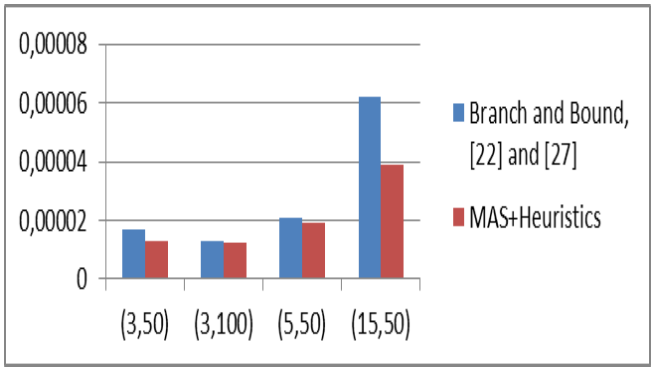

Figure 12. Results of the big size instance

Table 2. Obtained results for the big size instances

		Mokotoff		**Dell'Amico**		**Our system**	
M	**N**	Time (in seconds)	Non resolved instances	Time (in seconds)	Non resolved instances	Time (in seconds)	Non resolved instances
3	20	0.12	0	0.000014	0	0.000015	0
3	50	0.10	0	0.000017	0	0.000013	0
3	100	0.22	0	0.000013	0	0.0000124	0
3	200	0.12	0	0.000020	0	0.000014	0
5	20	0.57	0	0.000753	0	0.00042	0
5	50	0.35	0	0.000021	0	0.000019	0
5	100	0.67	0	0.000026	0	0.000022	0
5	200	1.23	0	0.000034	0	0.000027	0
15	20	0.01	0	0.000004	0	0.000025	0
15	50	864.47	2	0.000062	0	0.000039	0
15	100	329.66	0	0.000059	0	0.000037	0
15	200	348.79	0	0.000104	0	0.000041	0
100	200	2042.08	0	0.058836	0	0.00078	0

(c) Interpretation of our results and those of E.Mokotoff and DellAmico

Indeed, for the values taken from the three classes of instances (large and small sizes), we note that our values are better than those of E Mokotoff. For the Dell'Amico values, only for the instances of the first and second class of the small size, the execution time of Dell' Amico and al. is generally better than ours. This can be explained as follow: Dell' Amico and al. used an exact method which solves the problem faster than our method for the small size instances because our heuristic method requires a procedure of communication and exchange of messages between agents which take the majority of the execution time in this case. The founded result using the same benchmarks shows the effectiveness of our system. This mainly affects the execution time of the big size instances. This is another major contribution of our work since we could show, from the numerical results, that multi-agent negotiation combined with heuristics can be used as an approach to solve complex problems (and not as a distribution tool), to significantly reduce the search space dimension and to provide satisfactory and close to optimal solutions.

2) Obtained results for the Adriana Alvim and Celso Ribeiro instances:

(a) Comparing the execution time

To solve the P||Cmax, Adriana c.f. Alvim and Celso C Ribeiro [24] used an hybrid heuristic approach which includes a tabu search strategy. They make the analogy with the BinPacking problem which consists in finding the minimum necessary number of bins m. Every bin has a capacity C to pack a set of articles. The sum of the articles weights in each bin should not exceed its capacity. This problem is closely related to our problem. They share the same decision problem, which consist in determining if all the items/tasks can be assigned to m bins/machines with a capacity/makespan equal to C. Indeed, their approach is composed of a series of procedures; each one has a specific step of pretreatment. But they share the same set of processes composed of the following phases: construction, redistribution and improvement. Tabu search is at the heart of the improvement phase, achieving the function of the reconstitution of the feasible solutions every time the construction phase produces a solution which violates the problem constraints. A feasible solution for the BP is a solution where the makespan does not exceed the capacity of the concerned bin. To test their work, the authors used four groups of instances for the BP problem and two groups of instances for the P||Cmax problem. We are interesting in the P||Cmax problem: they used two groups of instances and they considered three possible intervals for the execution times Pi: [1, 100], [1, 1000] and [1, 10000]. The number of machines m is in {5, 10, and 25} and the number of tasks n is in {10, 50, 100, 500, and 1000}. These two groups of instances differ in the distribution of the execution time: either uniform or non-uniform. According to them, their code is better, in time, than the Branch & Bound of Dell'Amico and Martello. Table 3 compares our results to those of Adriana for the tested instances.

Table 3. Comparison of results

Pi	M	N	Our system	Adriana, 2010
1-1000	5	100	0.000022	0,01
1-10000	5	10	0.000036	0,22
1-10000	5	50	0.000019	0,15
1-10000	5	100	0.000022	0,01

(b) Comparing the makespan

We also compare, in table 4, the founded makespans to those of the Adriana's instances.

Table 4. Obtained Execution Times Compared to the Adriana and Celso Instances

Pi	N	M	H_CNP*	LPT*	C(B&B)*	LB_Adr*	Cmax_ Adr*
1-1000	5	100	10237	10247	10237	10237	10237
1-10000	5	10	11385	11575	11575	11385	11575
1-10000	5	50	55528	55747	55530	55528	55528
1-10000	5	100	96926	96960	96926	96926	96926

***H_CNP** : makespan of our system: CNP + Heuristics.
***LPT** : makespan obtained by the LPT heuristic.
***C(B&B)** : makespan obtained by the Brunch and Bound algorithm.
***LB_Adr** : lower bound found by the HI PCmax Adriana heuristic.
***Cmax_Adr**: makespan obtained by the HI PCmax heuristic of Adriana.

(c) Interpretations of the obtained results and those of Adriana Alvim and Celso

Thus, on the level of the execution times, our results are better than those of DellAmico and Martello and those of Adriana. Concerning the makespan, our results are similar to those of Adriana for the majority of the tested instances. These results, as mentioned in our previous works ([29] and [30]), are better than those founded by the LPT heuristic and generally better than those founded by the B&B algorithm.

V. Conclusion and Perspectives

The execution of parallel tasks generates a set of algorithmic and optimization problems. Due to their complexity, these problems cannot be resolved in a reasonable time.

Our work deals about the problem of allocating berths to vessels (the Berth Allocation Problem) in its dynamic and discrete variant noted DDBAP. To resolve this problem, we adopted a heuristic approach and an agent architecture which allows the interaction between intelligent agents to reach a common goal, which is the obtaining of a feasible and close to optimal solution. We use the OMaSE methodology and the tool agentTools, we provide a system modeling by establishing the OMaSE models. We developed our model of negotiation using the Jade platform. In conclusion, we obtained a good result that shows the effectiveness of the distributed heuristic method that combines the two concepts of 'Agent' and 'Heuristics'. Possible extensions of our work are numerous. We propose the following few prospects to improve the performance of our work:

- Taking into account other constraints such as precedence which affects the flow of container exchange between vessels and the integration of metaheuristics which proved their effectiveness in solving transport problems such as genetic algorithms and simulated annealing. We can also integrate hybrid methods.

– Improving our approach to treat other extensions of the BAP and adopting our model to solve the *Quay Crane Scheduling Problem* simultaneously with the *Berth Allocation Problem*. This problem which combines the two previous problems is called the *Berth and Quay Crane Allocation Problem*.

– Extending our system to support other real parameters and factors to treat unforeseen ones and to support disturbances (empty containers, delay and change of weather and arrival times, disaster management).

References

[1] I. Vacca, "Container Terminal Management: Integrated Models and Large-Scale Optimization Algorithms," Thèse EPFL no 4926, Ecole Polytechnique Fédérale de Lausanne, France, 2011. URL : http://library.epfl.ch/theses/?nr=4926

[2] J.F. Cordeau, G. Laporte, Pasquale, Legato and L. Moccia,"Models and Tabu Search Heuristics for the Berth Allocation Problem," Journal Transportation Science archive Volume 39 Issue 4, pp 526-538, 2005.

[3] G. Giallombardo, L. Moccia, M. Salani and I. Vacca, "The tactical berth allocation problem with quay crane assignment and transshipment-related quadratic yard costs," European Transport Conference (ETC); PROCEEDINGS, 2008.

[4] C. Bierwirth, F. Meizel, "A survey of berth allocation and quay crane scheduling problems in container terminals," European Journal of Operational Research 202 (2010), 615-627.

[5] A. Imai, E. Nishimura, S. Papadimitriou, "The dynamic berth allocation problem for a container port," Transportation Research,Vol 35, Issue 4, , pp 401-417, 2001.

[6] E. Nishimura, A. Imai., S. Papadimitriou, "Berth allocation planning in the public berth system by genetic algorithms," European Journal of Operational Research, v. 131, 282-292, 2001.

[7] Y. Guan, RK. Cheung, "The berth allocation problem: models and solution methods,". OR Spectrum 1: 75-92, 2004.

[8] J.-F. Cordeau, G. Laporte, and A. Mercier, "Improved Tabu Search Algorithm for the Handling of Route Duration Constraints in Vehicle Routing Problems with Time Windows," Journal of Operational Research Soc, 55:542 -546, 2004.

[9] A. Imai, E. Nishimura and S. Papadimitriou, " Berthing ships at a multi-user container terminal with a limited quay capacity," Transportation Research Part E, 44(1), 136-151,2008.

[10] P. Hansen, C. Oguz, and N. Mladenovic, "Variable neighborhood search for minimum cost berth allocation," European Journal of Operational Research, 191:636 – 649, 2008.

[11] M.M. Golias, M. Boile, and S. Theofanis, "An adaptive time window partitioning based algorithm for the discrete and dynamic berth scheduling problem," Transportation Research Record. (ISSN: 0361-1981), 2009b.

[12] S. Theofanis, M. Boile, and M.M. Golias, "Container terminal berth planning: critical review of research approaches and practical challenges" Transportation Research Record. (ISSN: 0361-1981), 2009.

[13] S. Mnasri, K. Zidi, "A heuristic approach based on the multi-agents negotiation for the resolution of the DDBAP," 4th International Conference on Metaheuristics and Nature Inspired Computing, Sousse - Tunisia, 27- 31 october 2012.

[14] Y. Ding, L. GuoLong, L. ChengJi, "Model and heuristic algorithm for Quay Crane Scheduling at container terminal," 9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), pp.1054-1061, Sichuan , china, 29-31 May 2012.

[15] X. Liang, W. Li, W. Zhao, B. Li, "Multistage collaborative scheduling of berth and quay crane based on heuristic strategies and particle swarm optimization," IEEE 16th International Conference on Computer Supported Cooperative Work in Design (CSCWD), pp.913-918, 23-25 May 2012.

[16] R. Zaghdoud, K. Zidi, K. Mesghouni, K. Ghedira, S. Collart-Ditullel, "Optimization Problem of Assignment Containers to AIVs in a Container Terminal," 13-th IFAC Symposium on Control in Transportation Systems, September 12 - 14, 2012, Sofia, Bulgaria.

[17] J.K. Lenstra, R. Kan and Brucker, "Complexity of machine scheduling problems," Annals of Discrete Mathematics,1, 343-362,1977.

[18] R.L. Graham, E.L. Lawler, J.K. Lenstra,"Optimization and approximation in deterministic sequencing and scheduling: a survey," Annals of Discrete Mathematics, vol. 5, p. 287-326,1979.

[19] N. Hashemian, "Makespan minimization for parallel machines scheduling with availability constraints," Computers and Operations Research Volume 36, Issue 6, June 2009, Pages 1809–1812.

[20] P.M. França, M. Gendreau, G. Laport and F. Muller, "A tabu search heuristic for the multiprocessor scheduling problem with sequence dependent setup times," International Journal of Production Economics, 43, 79-89, 1996.

[21] F. Antonio, N. Emiliano, S. MariaGrazia, "A Multi-Exchange Neighborhood for Minimum Makespan Parallel Machine Scheduling Problems," Journal of Combinatorial Optimization, 8(2):195-220, 2003.

[22] M. Dell'Amico and S. Martello, "A note on Exact Algorithms for the Identical Parallel Machine Scheduling Problem," European Journal of Operational Research,576-578, 2005.

[23] M. Iori, S. Martello, "Scatter Search Algorithms for Identical Parallel Machine Scheduling Problems," F. Xhafa and A. Abraham (eds.). Metaheuristics for Scheduling in Industrial and Manufacturing Applications, Springer, Berlin, 41–59, 2008.

[24] A.C.F. Alvim and C.C. Ribeiro, "A Hybrid Heuristic for Bin-Packing and Multiprocessor Scheduling," Journal of Heuristics.PROCEEDINGS, 2010.

[25] Meriam Kefi Gazdar,"Optimisation Heuristique Distribuée du Problème de Stockage de Conteneurs dans un Port,' Thèse de l'Ecole Centrale de Lille (en co-tutelle avec l'Ecole Nationale des Sciences de l'Informatique de Tunis), 2008.

[26] A. Scott, DeLoach, J. C. Garcia-Ojeda,"O-MaSE : a customizable approach to designing and building complex, adaptive multi-agent systems," International Journal of Agent-Oriented Software Engineering - Vol. 4, No.3 pp. 244 – 280,2010.

[27] M. Dell Amico and S. Martello, "Optimal scheduling of tasks on identical parallel processors," INFORMS Journal on Computing, 191 -200, ISSN 1091-9856, (was published as ORSA Journal on Computing from 1989 to 1995 under ISSN 0899-1499), 1995.

[28] E. Mokotoff, "An exact algorithm for the Identical Parallel Machine Sheduling Problem," European Journal of Operational Research, Vol 152, Issue 3, Pages 758–769, 2004.

[29] S. Mnasri, K. Zidi and K. Ghedira," A heuristic multi-agents model to solve the P||Cmax: application to the DDBAP," 13th International Conference on Hybrid Intelligent Systems, Hammamet-Tunisie, 04- 06 decembre 2013.

[30] S. Mnasri and K. Zidi,"A heuristic approach based on the multi-agents negotiation to resolve the DDBAP,"4th International Conference on Metaheuristics and Nature Inspired Computing,Tunisia, 2012.

Author Biographies

Sami Mnasri received his Engineering degree in Computer Science (speciality: Software Engineering.) from ESSTT University, Tunis, Tunisia in 2008, Master of Research in Computer Science (Speciality: Information Systems and New Technologies) from FSEGS University, Sfax, Tunisia in 2012 and he is now a PhD student in Computer Science in IRIT, Toulouse, France. He is currently an assistant professor in the Department of Computer Science at the FSG University, Gafsa, Tunisia. His research interests include deployment strategies, coverage, connectivity and energy optimisation issues in wireless sensor networks.

Nejah Nasri received his Engineering degree in Computer Science from ENIS University, Sfax, Tunisia in 2002, Master of Research in Computer Science from ENIS University, Sfax, Tunisia in 2004 and PhD in Computer Science from Université de Toulouse le Mirail II, Toulouse, France in 2010. He is currently a HDR professor in, (and the director of) the Department of Computer Science of the FSG University, Gafsa, Tunisia. His research interests include wireless sensor networks engineering, efficient energy MAC protocols, synchronisation, localization and performance analysis in wireless sensor networks.