Dynamic assignment problem of parking slots

Mustapha Ratli*§, Abdessamad Ait El Cadi†§, Bassem Jarboui‡¶, Abdelhakim Artiba‖§
§*Polytechnic University of Hauts-de-France*
LAMIH, UMR CNRS 8201, 59313 Valenciennes Cedex 9, France.
¶*M.O.D.I.L.S, Faculté des Sciences Economiques et de Gestion, Sfax, Tunisie*.
Email:* mustapha.ratli@uphf.fr
†abdessamad.aitelcadi@uphf.fr
‡bassem_jarboui@yahoo.fr
‖Abdelhakim.artiba@uphf.fr

***Abstract*—The parking problem has become one of the major issues in urban transportation planning and traffic management research. The present paper deals with the dynamic assignment problem of the parking slots (places). The objectives are to provide a global satisfaction of all customers and maximize parking occupancy. A dynamic assignment problem consists in solving a sequence of assignment problems over time. To cope with all these aspects, we offer in this paper, a new approach based on a learning strategy: an Estimation of Distribution Algorithm (EDA) where a reinforcement learning method is used to support the assignment algorithm. We tested our approach with simulations for over 120 days, using a set of up to 10 parking lots (i.e. with up to 7,000 parking slots) and 13,000 requests distributed with different patterns over a day. The comparative study between an assignment algorithm with and without reinforcement learning algorithm has proven the relevance of our approach: the saving is up to 80%. The results also showed the effects and the benefits of the learning strategy.**

***Keywords*—Assignment, dynamic, parking slots, meta-heuristic, learning, EDA.**

I. Introduction

A dynamic assignment problem consists in solving a sequence of assignment problems over time. At each time period, decisions must be made about what resources and tasks to assign to each other. Assignments that are made at earlier time periods affect the assignments that are made during later time periods, and information about the future is often uncertain. Some examples of dynamic assignment problems include dispatching truck drivers to deliver loads, pairing locomotives with trains and assigning company employees to jobs [1]. [2] propose a "smart parking" system for an urban environment based on a dynamic resource allocation approach. The goal of the system is to provide an optimal assignment of users (drivers) to the parking slots, respecting the overall parking capacity. The quality of an assignment is measured using a function that combines proximity to destination and parking cost. At each decision point, the system considers two queues of users: WAIT queue (consists of users who wait to be assigned to resources) and RESERVE queue (consists of users who have already been assigned and have reserved a resource in some earlier decision point). An optimal allocation of all users in both WAIT queue and RESERVE queue at each decision point is determined by solving a mixed integer linear programming problem. Simulation results show that the "smart parking" approach provides near-optimal resource utilization and significant improvement compared to classical guidance-based systems. In [3], the authors present an approach for assigning parking slots using a semi-centralized structure that uses Parking Coordinators (PC). They propose two variants: in the first variant, each PC affects a parking independently of the others; in the second variant, the PCs interact with their near neighbors, to assign a location with the constraint of balancing the load of each car park. The idea behind this approach is to use PCs to gather vehicles (drivers)' queries during a certain time window and assign a parking slot according to these drivers preferences.

II. Mixed integer programming formulation

A driver represents a user and a parking slot represents a resource. A driver $i \in I = \{1, 2, \ldots, n\}$, aiming to visit a given destination, starts looking for a parking slot by launching a request at a non–deterministic moment. These moments are not known in advance and are discovered during the assignment process. A parking $j \in J = \{1, 2, ...m\}$ has a certain number of available slots that change during the time. Both drivers' requests and available parking slots appear in a non–deterministic way and can change their state at any moment. We assume that each resource, each driver and each destination has a known location associated to it in a two–dimensional Euclidean space. The density of drivers requests is not uniform, it varies from a time period to another during the same day. For example, the requests density for a parking slot near a hospital rises in visiting hours and declines at other times. As a consequence, to take into account this feature, each day of the time horizon is subdivided into a fixed number T of equally spaced time periods. Periods that represent peak hours, normal hours, ... off-peak hours. The objective of assigning vehicles to parking slots is to provide a global satisfaction of all customers. For instance, when the decision is performed request per request independently using a FIFO (First In First Out) rule, it may not be satisfactory because it can have a negative impact on the future situations. The following example, presented in Figure 1, illustrates the impact of the starting choice on the future decision when using a FIFO rule. We assume that we have one available parking slot in parking p_1 and another in parking p_2. Moreover, we

978-1-7281-1566-5/19/$31.00 © 2019 IEEE

suppose that vehicle v_1 launches a request for a parking slot near destination d_1 and vehicle v_2 looks for a place near destination d_2. In addition, it is assumed that vehicle v_1 arrives before vehicle v_2 into the system. If we aim to minimize the distance between the parking and the destination, the FIFO rule will suggest to assign vehicle v_1 to parking p_1 and vehicle v_2 to parking p_2. As the vehicle v_1' request is handled first, regardless of the vehicle v_2' request. The result is "good" for deriver v_1 but "bad" for driver v_2 and bad for the global system. This is due to the fact that the future information concerning the second driver is excluded. However, if vehicle v_1 is assigned to parking p_2 and vehicle v_2 to parking p_1, driver v_1 preserves his satisfaction degree as the distance from parking p_1 to destination d_1 is similar to the distance from parking p_2 to destination d_1, but the degree of satisfaction of driver v_2 is increased.

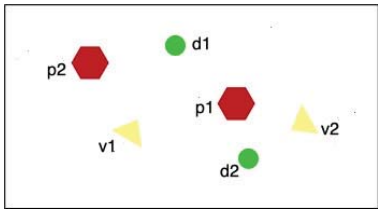

Fig. 1. Illustrative example

To prevent this drawback, two alternatives may be presented: the first one consists in collecting a subset of the requests in a given moment and solving the associated problem. The second one consists in establishing a forecast process to model the future information. In this section, we will discuss the first alternative and in a later section we will propose a forecast approach. The operation of collecting requests needs a collecting time period. We define a partition of time to gather requests and to determine the number of available slots in each parking created by the outgoing vehicles. Each time period, $t = 1, 2, ..., T$, of a day is partitioned into fixed K equally spaced sub-periods. We denote by k the index of the k^{th} sub-period. The process of assignment is performed in a discrete given moment, at the end of each sub-period called "decision point". Each decision point has also an index denoted by k.

At each decision point k, we denote by N_k the set of new requests gathered during the sub-period k. Due to the limited capacity of the parking, some requests may not be satisfied at each decision point k. We denote by R_k the set of yet unassigned vehicles at the decision point k. Therefore, the set of vehicles that have to be assigned at the next decision point $k+1$ is $E_{k+1} = N_{k+1} \cup R_k$, the sum of new requests N_{k+1} and the non-handled requests up-to-now R_k. To formally define the dynamic assignment problem, we need the following:

1) Parameters:
   - $dest_i^k$: location of the destination of vehicle i from the set E_k.
   - loc_i^k: location of vehicle i from the set E_k.
   - $d_{i,j}^k$: distance between the location of the vehicle i from the set E_k and the parking j at the decision point k.
   - $d_{[i],j}^k$: distance between the destination $dest_i^k$ of the vehicle i from the set E_k and the parking j at the decision point k.
   - V: the velocity of vehicles.
   - V': the walking velocity of a driver needed to reach the destination after parking the vehicle.
   - w_i^k: the waiting time of a vehicle i between the launch of the request and the decision point k.
   - C_j^k: the capacity of the parking j at the decision point k.
2) Variables: variables defined as follows:

$$x_{i,j}^k = \begin{cases} 1 & \text{if the vehicle } i \text{ is assigned to parking } j \\ & \text{at the decision point } k, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

3) Criteria: as the occupancy time of a parking slot begins from the assignment of the requests, which is not necessarily the arrival time at the parking, the objective of the decision maker is to minimize the total distance between all vehicles and their assigned parking slots. This maximizes the occupancy and satisfies the parking manager. However, to guarantee a good quality of service to the customers, it is also necessary to minimize two elements:
   - the distance traveled between the assigned parking slot and the customer's final destination.
   - the waiting time that separates the launch of the request and the decision.

   These objectives are aggregated into a single weighted objective as follows:

$$Minf^k(x) = \sum_{i \in I} \sum_{j \in J} x_{i,j}^k (\lambda_1 \frac{d_{i,j}^k}{V} + \lambda_2 \frac{d_{[i],j}^k}{V'} - \lambda_3 w_i^k) \tag{2}$$

   where λ_1, λ_2 and λ_3 are non-negative values denoting the weights of each criterion. The non-positive coefficient associated to the third criterion is used as a priority factor.
4) Constraints: the main constraints in the considered problem at each decision point deal with the limited capacity of each parking and the satisfaction of the drivers' requests. However, as the total number of available parking slots at each decision point is not necessarily equal to the total requests to be assigned, the constraints will be different.Three possible cases will be considered.
   - **Case 1:** The number of available slots is larger than the number of requests at the k^{th} decision point:

$$\sum_{i \in I} x_{i,j}^k \leq C_j^k \qquad \forall j \in J \tag{3}$$

$$\sum_{j \in J} x_{i,j}^k = 1 \qquad \forall i \in I \tag{4}$$

   Constraints (3) guarantee that, at each decision point k, the number of assigned cars in the parking j

cannot exceed its capacity. Constraints (4) ensure that each driver i is assigned to one and only one parking slot.

- **Case 2:** The number of available slots is less than the number of requests at the k^{th} decision point:

$$\sum_{i \in I} x_{i,j}^k = C_j^k \qquad \forall j \in J \tag{5}$$

$$\sum_{j \in J} x_{i,j}^k \leq 1 \qquad \forall i \in I \tag{6}$$

In this case, we formulate the constraints such that all the slots in the parking j will be occupied and some drivers may not be assigned to any parking slot.

- **Case 3:** The number of available slots is equal to the number of requests at the k^{th} decision point:

$$\sum_{i \in I} x_{i,j}^k = C_j^k \qquad \forall j \in J \tag{7}$$

$$\sum_{j \in J} x_{i,j}^k = 1 \qquad \forall i \in I \tag{8}$$

In this last case, the offer of the parking slots is equal to the demand and thus each parking slot should be occupied and each driver should find a parking slot.

To make the situation clearer, we propose to formulate our problem as a simple assignment problem, where the capacity of each agent is equal to one. This is done by disaggregating the parking slots. Instead of considering the slots through the capacity of each parking we consider the available slots individually. Consequently, the problem becomes an assignment of drivers to parking slots instead of an assignment of drivers to parking. Let J_j^k be the set of available slots in a parking j in the set J at the decision point k. The binary decision variables will be defined, from now on, as follows:

$$x_{i,h}^k = \begin{cases} 1 & \text{if the vehicle } i \text{ is assigned to parking} \\ & \text{slot } h \text{ at the decision point } k, \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

The new proposed formulation of our problem can be presented as follows:

$$\begin{aligned} Minf^k(x) = \quad & \sum_{i=1}^{n} \sum_{j \in J} \sum_{h \in J_j^k} \\ & x_{i,h}^k (\lambda_1 \tfrac{d_{i,h}^k}{V} + \lambda_2 \tfrac{d_{[i],h}^k}{V'} - \lambda_3 w_i^k), \end{aligned} \tag{10}$$

subject to the following constraints:

- **Case 1:** The number of available slots is higher than the number of requests at the k^{th} decision point:

$$\sum_{i \in I} x_{i,h}^k \leq 1 \qquad \forall j \in J, \forall h \in J_j^k \tag{11}$$

$$\sum_{j \in J} \sum_{h \in J_j^k} x_{i,h}^k = 1 \qquad \forall i \in I \tag{12}$$

- **Case 2:** The number of available slots is less than the number of requests at the k^{th} decision point:

$$\sum_{i \in I} x_{i,h}^k = 1 \qquad \forall j \in J, \forall h \in J_j^k \tag{13}$$

$$\sum_{j \in J} \sum_{h \in J_j^k} x_{i,h}^k \leq 1 \qquad \forall i \in I \tag{14}$$

- **Case 3:** The number of available slots is equal to the number of requests at the k^{th} decision point:

$$\sum_{i \in I} x_{i,h}^k = 1 \qquad \forall j \in J, \forall h \in J_j^k \tag{15}$$

$$\sum_{j \in J} \sum_{h \in J_j^k} x_{i,h}^k = 1 \qquad \forall i \in I \tag{16}$$

The considered problem is dynamic because, during a given day, the demand for parking slots is not uniform and the number of requests launched by the drivers changes from one period $t \in T$ of the day to an other. Therefore, assignments that are made at earlier periods affect which assignments can be made during later periods. Moreover, the information about the future periods is uncertain. Our goal is to efficiently manage the requests while dealing with this uncertainty. To do so, we propose to establish a forecasting process based on a learning effect. We introduce a penalty term in the objective function that determines for each parking and each period of the day whether or not the current assignment has an impact on the future ones or not. In other words, if the future demand around a parking j is going to be higher, the current penalty should be big, in order to leave more available slots in this parking for the future period. Otherwise, the penalty will be small, in order to make the slots of this parking more attractive for the current assignment. The value of these penalties are calibrated through a learning process.

Let p_j^t denote the penalty term associated to parking j at the period of time t $(t = 1, 2, ..., T)$. It should be noted that for each slot in the parking j, the penalty is equal to the parking penalty p_j^t. In addition, between two consecutive time periods t_1 and t_2, such that $t_1 < t_2$ and for a decision point k such that $t_1 \leq k < t_2$, we set $p_j^k = p_j^{t_1}$.

The new objective function can be written as follows:

$$\begin{aligned} Minf^k(x) = \quad & \sum_{i=1}^{n} \sum_{j \in J} \sum_{h \in J_j^k} \\ & x_{i,h}^k (\lambda_1 \tfrac{d_{i,h}^k}{V} + \lambda_2 \tfrac{d_{[i],h}^k}{V'} - \lambda_3 w_i^k + p_j^k) \end{aligned} \tag{17}$$

III. Estimation of distribution algorithms

The EDA was first introduced by [4], and it is a stochastic optimization technique that explores the space of potential solutions by building and sampling explicit probabilistic models of promising candidate solutions. Starting with a population of

individuals (candidate solutions), generally randomly generated, this algorithm selects good individuals regarding fitness. Then, a new distribution of probability is estimated from the selected candidates. Next, new offsprings are generated from the estimated distribution. The process is repeated until the termination criterion is met.

Algorithm 1: Basic EDA

Generate an initial population of P individuals
repeat
  Select a set of Q parents from the current population P with a selection method;
  Build a probabilistic model for the set of selected parents;
  Create a new offspring to P according to the estimated probability distribution;
  Replace some individuals in the current population P with new individuals;
until *a stopping criterion is met*;

The choice of the probabilistic model is not a trivial task. Many works have focused on the way to establish the distribution of probability that allows to capture the features of the promising solutions. Three classes of EDA may be presented according to the chosen probabilistic model and the degree of dependency between the variables of the problem. The first class called *univariate model*, assumes no dependencies between variables of candidate solutions, that is to say that all variables are independent. The second one, called *bivariate model*, assumes only pairwise dependencies between these variables and the last class, called *multivariate model*, assumes multiple dependencies between variables.

1) Univariate marginal distribution algorithm: This algorithm, proposed by [5], behaves differently from the two previous algorithms. It estimates the joint probability distribution $p^g(x)$ of the selected individuals at each generation. In the case of binary variables, the probability vector is estimated from marginal frequencies and $p^g(x_i)$ is set by counting the number of occurrences of "1", $f_k(x_i = 1)$ for $k = 1, 2, ..., Q$, in the set of selected individuals. In order to generate new individuals, each variable is generated according to $p^g(x_i)$ as follows

$$p^g(x_i) = \frac{1}{Q} \sum_{k=1}^{Q} f_k(x_i = 1) \tag{18}$$

For continuous domains, the Univariate Marginal Distribution Algorithm is designed through statistical tests to find the density function that best fits the variables. Then, using the maximum likelihood estimates, the evaluation of the parameters is performed.

A. Bivariate models

In order to make the interactions between variables more realistic, this class of models takes into account pairwise dependencies. In this class of EDA, we focus only on the Mutual Information Maximization for Input Clustering (MIMIC) proposed by [6] as it is used for both continuous and discrete domains. In MIMIC, the conditional dependencies of $p^g(x)$ are defined by a Markovian chain in which each variable is conditioned by the previous one. Therefore, in each generation, the MIMIC uses a permutation framework of ordered pairwise conditional probabilities, which can be written as follows:

$$p^g_\pi(x) = \quad p^g(x_{i_1}|x_{i_2})p^g(x_{i_2}|x_{i_3}) \quad \dots p^g(x_{i_{n-1}}|x_{i_n}) \tag{19}$$

where $\pi = \{i_1, i_2, ..., i_n\}$ is a permutation of the indexes $1, 2, ..., n$. The objective is to find the best $p^g_\pi(x)$ as close as possible to the complete joint probability:

$$p^g(x) = \qquad p^g(x_1|x_2, \dots, x_n) \qquad p^g(x_2|x_3, \dots, x_n) \dots p^g(x_{n-1}|x_n)p^g(x_n) \tag{20}$$

The degree of similarity between $p^g_\pi(x)$ and $p^g(x)$ is measured bu using the Kullback-Leibler distance.

B. Multivariate models

This section discusses the models that do not impose any restriction about the dependencies among variables.

IV. Estimation of distribution algorithm with reinforcement learning

Generally speaking, the estimation of distribution algorithm proceeds as follows. First, an initial population of candidate solutions is generated randomly. Then, a subset of solutions is selected from the initial population. At this moment comes the main step of the algorithm, consisting of building a probability model based on the selected solutions to create a new "good" solution. Finally, the step of replacement decides if the new solution should be kept or not. The algorithm continues until it reaches a stopping criterion. In our context, we propose to apply the EDA to find good values for the penalties p^k_j.

Therefore, the problem considered in this section consists of finding the best values for the matrix of the the penalties p^k_j to be used in our dynamic assignment problem. This current problem will be referred as the Penalties Calibration Problem (PCP). Our proposed algorithm follows these steps:

1) Encoding solution: A solution of the PCP problem is encoded by a matrix π where the rows represent the parking $j \in J$ and the columns correspond to the time periods $t = 1, 2..., T$. The intersection between each row j and each column t represents the penalty term p^t_j. Figure(2).

$$\pi = \begin{pmatrix} p^1_1 & p^2_1 & ... & p^T_1 \\ p^1_2 & p^2_2 & ... & p^T_2 \\ : & : & ... & : \\ : & : & ... & : \\ p^1_m & p^2_m & ... & p^T_m \end{pmatrix}$$

Fig. 2. Encoding solution

2) Evaluation and forecasting: Each solution of the PCP problem is evaluated according to the objective function

described in (17). The evaluation of any given solution of the PCP is the total assignment cost, over a day, for the dynamic assignment problem with the corresponding penalties. That is to say, a solution of the PCP is a set of penalties. These penalties are used in equation (17); the problem is solved for all the decision points, the total cost of period t is given by $f^t(x) = \sum_{k=1}^{K} f^k(x)$ and the total cost of a day d is $f^d(x) = \sum_{t=1}^{T} f^t(x)$. After that, a smoothing technique is used to take into account the forecasting of the demand. At the end, the evaluation of any solution π of PCP is given by the following equation:

$$F(x) = \sum_{q=0}^{\delta} \alpha(1-\alpha)^q f^{d-q}(x), \quad (21)$$

where $0 < \alpha < 1$ is the smoothing factor which represents the weight of the previous observations. Therefore, the EDA consists to minimize $F(x)$.

3) Initial population: The initial population of P solutions is randomly generated. This means that we generate P matrices π such that each penalty $p^t_{j,r}$ (penalty for parking j during period t in the PCP solution r) of matrix π_r is generated according to a uniform distribution.
4) Selection: From the initial population we propose to select Q solutions according to the ranking of the objective functions $F(x)$ defined in equation (21).
5) Probabilistic model: In order to generate new candidate solutions, in our proposition we use the probabilistic model of the univariate marginal distribution algorithm for Gaussian models [7] where the parameters mean and standard deviation of a solution are extracted from population information during the optimization process. The two parameters to be estimated at each generation for each variable are the mean $\hat{\mu}_j$ and the standard deviation $\hat{\sigma}_j$. Their respective maximum likelihood estimates are:

$$\hat{\mu}^t_j = \bar{p^t_j} = \frac{1}{Q}\sum_{r=1}^{Q} p^t_{j,r} \quad (22)$$

$$\hat{\sigma}^t_j = \sqrt{\frac{1}{Q}\sum_{r=1}^{Q}(p^t_{j,r} - \bar{p^t_j})^2} \quad (23)$$

6) Replacement: We compare the new solution by the worst solution in the current population. If the new solution is better than this solution, then the worst solution is removed from the population and it is replaced with the new one.
7) Stopping criterion: The stopping criterion indicates when the search finishes. We set a maximum number of iterations and a maximal computational time in our algorithm.
8) Local search procedure for dynamic assignment problem: The local search procedure was proposed to improve the performance of the algorithm through problem decomposition. Instead of tackling the whole complex assignment problem at the same time, the problem is divided into a set of smaller sub-problems, each of which can be solved easily in terms of computational time. The purpose of the decomposition scheme is to break down a large problem into smaller ones. Practically, for small assignment problems or when we have enough time, we use an adaptation of the Munkres' Assignment Algorithm, but when the size is huge a local search procedure is used. In fact, if the number of vehicles that appear in the system and the number of considered parking slots are large, then the number of variables and constraints taken into account for solving the whole problem at each decision point may be huge. The idea is to decompose the area of the system (city) into a set of regions and solve an assignment problem for each of them. For each iteration, a region is selected randomly, we record the set of requests Ω from vehicles in this region and the set of parking lots L existing in the same region.Then, the problem formulated by the associated variables is solved while taking into account the assignments of the remaining regions. Therefore, we set a threshold on the number of vehicles n_{max} from which the local search procedure is applied. The procedure starts from an initial solution randomly generated. Then, we select at random some decision variables to be fixed and optimize the remaining sub-problem, according to the objective function. The process is repeated until a given stopping criterion is reached (Algorithm 2). The framework of the proposed algorithm with forecast is given in Figure 3 and Algorithm 3. It should be noted that if we set π_{best} to 0, we obtain the assignment algorithm without forecasting process. We denote by AA_{EDA} and AA the assignment algorithm with and without EDA, respectively.

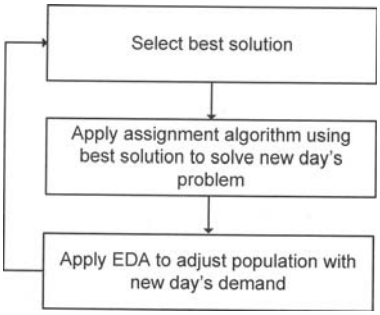

Fig. 3. The proposed algorithm

V. Conclusion

We approached the problem of dynamic assignment for parking slots. A driver aiming to goto a given destination starts looking for a parking slot by launching a request at a non-deterministic moment. The parking lot manager has to fulfill these requests by assigning available slots to vehicles. The objectives are to provide a global satisfaction to all customers and to maximize the parking lots occupancy. The problem is dynamic, the requests and the parking lots change over time.

Algorithm 2: Pseudo-code of the local search procedure

repeat
  Select a set L of parking slots at random at the decision point k;
  Find the set Ω associated to those spaces at the decision point k;
  Solve the assignment problem of (Ω, L, π_{best});
until *A stopping criterion is met*;

Algorithm 3: Pseudo-code of the assignment algorithm with forecast process based on EDA

$R_0 = \oslash$;
for $k = 1, 2, ..., (K \times T \times D)$ **do**
  Find N_k, R_k and loc_i^k;
  $E_k = \{N_k \bigcup R_{k-1}\}$;
  if $|E_k| < n_{max}$ **then**
    (A_k, R_k)=Apply assignment algorithm $\left(E_k, J_j^k, \pi_{best}\right)$;
  else
    (A_k, R_k)=Apply local search procedure $\left(E_k, J_j^k, \pi_{best}\right)$;
  Update loc_i^k of R_k

Firstly the problem is modeled as a sequence of consecutive assignment problems, over time. These problems are interrelated. At each decision point (a small time window) a static assignment problem is solved, we assign non-handled requests up-to-this point to the current available slots. Secondly as the assignments that are made at earlier periods affect which assignments can be made during later periods, we propose to establish a forecasting process based on a learning effect. We introduce penalty terms in the objective function. The values of these penalties are calibrated through a learning process using the Estimation of Distribution Algorithm (EDA). We notice that solving each assignment problem at every decision point can be time consuming depending on the number of the concerned requests and available slots. A local search procedure is proposed to improve the performance of our approach through problem decomposition and, consequently, to reduce the solving time. Instead of tackling the whole complex assignment problem at the same time, the problem is divided into a set of smaller sub-problems. We tested our approach with and without the learning effect. Our approach is efficient since we were able to manage a set of parking lots, of up-to 10 parking lots, during a horizon of 120 days, which corresponds to assignment problems with up-to 7000 parking slots to manage and 13000 requests per day to handle. The results also show the benefit of the learning effect. The total cost of the solutions with learning effect is less than the cost of the solutions without learning effect (a student test is used to prove the difference between these two methods).

REFERENCES

[1] M. Z. Spivey and W. B. Powell, "Some fixed-point results for the dynamic assignment problem." *Annals OR*, vol. 124, no. 1-4, pp. 15–33, 2003.

[2] Y. Geng and C. G. Cassandras, "Dynamic resource allocation in urban settings: A smart parking approach," in *Computer-Aided Control System Design (CACSD), 2011 IEEE International Symposium on*. IEEE, 2011, pp. 1–6.

[3] N. Mejri, M. Ayari, and F. Kamoun, "An efficient cooperative parking slot assignment solution," in *UBICOMM 2013, The Seventh International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, 2013, pp. 119–125.

[4] H. Mühlenbein and G. Paass, "From recombination of genes to the estimation of distributions i. binary parameters," in *Proceedings of the 4th International Conference on Parallel Problem Solving from Nature*, ser. PPSN IV. London, UK, UK: Springer-Verlag, 1996, pp. 178–187. [Online]. Available: http://dl.acm.org/citation.cfm?id=645823.670694

[5] H. Mühlenbein, "The equation for response to selection and its use for prediction," *Evol. Comput.*, vol. 5, no. 3, pp. 303–346, 1997.

[6] J. S. D. Bonet, C. L. Isbell, Jr., and P. Viola, "Mimic: Finding optima by estimating probability densities," in *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*. The MIT Press, 1996, p. 424.

[7] P. Larrañaga and J. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Boston, MA: Kluwer, 2002.

[8] Q. Zhang, J. Sun, E. Tsang, and J. Ford, "Estimation of distribution algorithm with 2-opt local search for the quadratic assignment problem," in *Towards a New Evolutionary Computation. Advances in Estimation of Distribution Algorithm*. Springer-Verlag, 2006, pp. 281–292.

[9] H. Li, Q. Zhang, E. Tsang, and J. Ford, "Hybrid Estimation of Distribution Algorithm for Multiobjective Knapsack Problem," in *Evolutionary Computation in Combinatorial Optimization*, 2004, pp. 145–154.

[10] T. Paul and H. Iba, "Linear and combinatorial optimizations by estimation of distribution algorithms," 2002.

[11] V. Robles, P. de Miguel, and P. Larrañaga, "Solving the traveling salesman problem with edas," in *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, 2002, pp. 211–229.

[12] S. Baluja, "Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning," Tech. Rep., 1994.

[13] M. Sebag and A. Ducoulombier, "Extending population-based incremental learning to continuous search spaces," in *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, ser. PPSN V. London, UK, UK: Springer-Verlag, 1998, pp. 418–427.

[14] S. Rudlof and M. Köppen, "Stochastic hill climbing with learning by vectors of normal distributions," 1996, pp. 60–70.

[15] P. Larrañaga, R. Etxeberria, J. A. Lozano, and J. M. Peña, "Optimization in continuous domains by learning and simulation of Gaussian networks," in *Proceedings of the 2000 Genetic and Evolutionary Computation Conference*, A. S. Wu, Ed., 2000, pp. 201–204.

[16] P. Larrañaga, J. A. Lozano, and E. Bengoetxea, "Estimation of distribution algorithms based on multivariate normal distributions and Gaussian networks," Dept. of Computer Science and Artificial Intelligence, University of Basque Country, Tech. Rep., 2001.

[17] P. A. Bosman and D. Thierens, "Linkage information processing in distribution estimation algorithms," in *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-1999*, W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, Eds., vol. I. Morgan Kaufmann Publishers, San Francisco, CA, 1999, pp. 60–67.

[18] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the society for industrial and applied mathematics*, vol. 5, no. 1, pp. 32–38, 1957.