Expert Systems with Applications 39 (2012) 316–327

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

A stochastic neighborhood search approach for airport gate assignment problem

Hakkı Murat Genç[a,b,*], Osman Kaan Erol[c], İbrahim Eksin[b], Mehmet Fatih Berber[d], Binnur Onaran Güleryüz[d]

[a] The National Research Institute of Electronics and Cryptology, Information Technologies Institute, TR-41470 Gebze, Kocaeli, Turkey
[b] İstanbul Technical University, Faculty of Electrical and Electronics Engineering, Control Engineering Department, Maslak, TR-34469 İstanbul, Turkey
[c] İstanbul Technical University, Faculty of Electrical and Electronics Engineering, Computer Engineering Department, Maslak, TR-34469 İstanbul, Turkey
[d] TAV Information Technologies Corporation, İstanbul Atatürk International Airport, Yeşilköy, TR-34149 İstanbul, Turkey

ARTICLE INFO

Keywords:
Gate assignment problem
Big Bang–Big Crunch optimization algorithm

ABSTRACT

An appropriate and efficient gate assignment is of great importance in airports since it plays a major role in the revenue obtained from the airport operations. In this study, we have focused mainly on maximum gate employment, or in other words minimize the total duration of un-gated flights. Here, we propose a method that combines the benefits of heuristic approaches with some stochastic approach instead of using a purely probabilistic approach to top-down solution of the problem. The heuristic approaches are usually used in order to provide a fast solution of the problem and later stochastic searches are used in order to ameliorate the previous results of the heuristic approach whenever possible. The proposed method generates an assignment order for the whole planes that corresponds to assignment priority. The ordering process is followed by the allocation step. Since, in practice, each airport has its own physical architecture, there have been arisen many constraints mainly concerning airplane types and parking lots in this step. Sequentially handling the plane ordering and allocation phases provides us great modularity in handling the constraints. The effectiveness of the proposed methodology has been tried to be illustrated firstly on fictively generated flight schedule data and secondly on the real world data obtained from a real world application developed for İstanbul Atatürk Airport.

1. Introduction

Gate assignment problem (GAP) has increasing importance due to the increasing passengers in airports. Though it is easy to understand the problem definition and meaning of the various objective functions derived in the literature, many studies covered the same problem with different approaches. The total number of solution candidates is number of planes over number of gates, and for a practical airport, this product yields impractical amounts of candidate solutions to be tried. So, the grandeur of the solution space and the existence of various and problem dependent constraints make the problem still difficult to solve for the optimum solution and therefore still up to date.

The lack of a precise ordering among the solutions such as in the case of a numerical parameter optimization problem makes the problem almost impossible or – at least – difficult to solve using any gradient search methodology. Since one cannot guarantee the order of a flight to gate assignment list to another one, it is also impossible to obtain an ‘obvious’ parameter-objective function graphics. Possible objective functions can be defined in terms of the staying time of the planes in the gates, number of passengers in aircrafts, the total walking distances belonging to the passengers of all scheduled flights within a specified and closed time interval. Therefore, the problem formulation can vary quite a lot due to this large span of objectives. Moreover, basic gate assignment problem is NP-hard (non-deterministic polynomial-time hard) (Obata, 1979) quadratic assignment problem. Because of these, there are various approaches to this problem in the literature with respect to requirements imposed. The gate assignment problems can be categorized with respect to objective functions, mathematical formulations, time slot models and constraint satisfaction strategies. Besides, the solution approaches have two heavily interacting main branches: rule based expert systems and mathematical models. In our implementation, the GAP objective is to maximize total gate time as an integer programing mathematical formulation that uses multiple time slots and the basic constraint that allows one flight at one gate at one time. No rule based expert system is utilized algorithm; but in the system developed for Atatürk Airport, the constraints are processed by user defined rules.

Teodorovic and Guberinic (1984) and Teodorovic and Stojkovic (1990) focus on total passenger delay and the number of flights

* Corresponding author at: The National Research Institute of Electronics and Cryptology, Information Technologies Institute, TR-41470 Gebze, Kocaeli, Turkey. Tel.: +90 262 6772624; fax: +90 262 6463187.
E-mail address: murat.genc@bte.tubitak.gov.tr (H.M. Genç).

cancellations in the case of irregularity of flights. Among other possible criteria, passenger walking distances (Babic, Teodorovic, & Tosic, 1984; Bandara & Wirasinghe, 1992; Ding, Lim, Rodrigues, & Zhu, 2004a, 2004b, 2005; Haghani & Chen, 1998; Hu & Paulo, 2007; Wei & Liu, 2007; Wirasinghe & Bandara, 1990); baggage transfer distances (Haghani & Chen, 1998; Hu & Paulo, 2007) are also considered. Chang (1994) considers the distance covered by passengers in carrying their baggage as an objective in addition to passenger walking distance. Even any objective criterion has factions in implementation: for example passenger walking distance can be handled as: (i) minimize the sum of total distance that all passengers walk, (ii) minimize the distance after baggage claim area, (iii) minimize connection flight travelling distance, (iv) minimize the maximum distance that a passenger need to walk, (v) minimize the number of passengers that need to walk more than $\boldsymbol{x}$ units. The list can be further extended. Unfortunately, assignment objectives depending on passenger walking distance are quite fragile (Dorndorf, Drexl, Nikulin, & Pesch, 2007).

Genetic algorithms (GAs) are the most known and widespread used global optimization methods. Since GAs use random number generators and they exhibit an ability to avoidance to get trapped to local optima they are considered to be successful search procedures when the objective function is nonlinear, non-derivative and discontinuous. Some researchers proposed GA based methods for the gate assignment problem (Bolat, 2001; Gu & Chung, 1999; Hu & Paulo, 2007). All the approaches utilizing population based routines, including GA based approaches, use global optimization methods to top down solve the problem or to improve the result of some heuristics. However, forming a complete solution candidate or altering the list once all the flights are assigned can be quite tardy for GA or similar GA like stochastic methods since they oblige to check all constraints to build up a valid solution. Therefore, using stochastic methods to ameliorate assignment after all the list has been built up is not a good solution alternative for GAP in practical applications.

For the Atatürk Airport's operator, criterion of highest priority is to increase the revenue obtained from the gate allocation operation. The most important parameter in the revenues is therefore the allocated gates, which are available in a limited number. The more efficiently the gates are assigned to the aircrafts, the lesser idle time is left between two successive flights and this means that more passengers use the gates. Hence, the revenue and passenger satisfaction are both increased. Flight gates are the major items addressed in the GAP. At İstanbul Atatürk Airport, as well as the most of the airports throughout the world, the revenues are majorly dependent upon assignment of an airplane to a gate or not. This leads to a cost function which changes greatly if an airplane is assigned to a gate or not. This gives rise to a discontinuous objective function or more generally, a cost or fitness function where intergate aircraft switches do not have a great influence on it. Furthermore, the gate number assigned to an aircraft has only an influence on passenger walking distances in linearly positioned parking lots but it has a minimal effect in airports with a star topology.

In this study, we propose a method that combines the benefits of heuristic approaches which provide once a fast initiating solution of the problem and later conduct stochastic searches in order to ameliorate the previous result obtained via heuristic approaches. Big Bang–Big Crunch (BB–BC) optimization algorithm (Erol & Eksin, 2006) is utilized for the stochastic neighborhood moves due fast convergence rate. The GAP formulation is detailed in Section 2 and BB–BC method is summarized in Section 3. The details of the new method are presented in Section 4 and the simulation results are given in Section 5. The İstanbul Atatürk Airport specific modifications and the developed system for this airport are discussed with the performance improvement reports in Section 6. The concluding remarks are finally given in Section 7.

2. Problem formulation

In this work, we approach the gate assignment problem as a maximum squeezing problem to the gates. In other words, the objective is to maximize gate duration, which is total time of the gates allocated for all flights of a day. This formulation generalizes the solution and reduces the dependence on the physical properties. Besides, it is the most important criterion by far among other reported performance measure as it is highly correlated with the revenue obtained. The basic constraint of the GAP imposed in the formulation can be stated as follows: one gate can only accommodate a single aircraft at a time and that therefore two flights must not be assigned to the same gate if their staying times overlap in time (Dorndorf et al. 2007). To measure density of the gates, the whole day is sampled for n minutes, where n can be chosen as 5 or 10 in a practical application. Note that selection of the length of a time slot directly effects the algorithm run time. In literature, selected time slots are in between five minutes and an hour duration (Bolat, 1999; Bolat, 2001; Haghani & Chen, 1998). We call this time interval corresponding to n minutes *a time slot* and the density is measured by counting up allocated timeslots.

The parameters related to gate assignment problem are defined as follows:

N	number of aircrafts
Ng	number of gates
Noa	number of open air parking places
Ns	number of stands where $Ns = Ng + Noa$
Nt	number of time slots in a day (depends on time slot length n, $Nt = 24 * 60/n$)
$T_{A(i)}$	arrival time of flight i
$T_{D(i)}$	departure time of flight i
M_u	($NxNt$) matrix of aircrafts (scheduling) where, $M_u(i,j) = 1$, if the aircraft i is at the airport in time slot j according to $T_{A(i)}$ and $T_{D(i)}$, $M_u(i,j) = 0$, if otherwise
M_c	($NsxNt$) matrix of assignments (gate assignments) where, $M_c(i,j) = U$, ($U = 1,\ldots,N$), if the gate i is assigned at time slot j to the Uth flight, $M_c(i,j) = 0$, if otherwise

The function to be maximized can be formulated as follows:

$$F_{\text{fitness}} = \sum_{k=1}^{Ng} \sum_{l=1}^{N_t} \text{any}(M_c(k,l)), \tag{1}$$

where,

$$\begin{aligned} &\text{any}(M_c(k,l)) = 1, \quad \text{if } M_c(k,l) \neq 0, \\ &\text{any}(M_c(k,l)) = 0, \quad \text{if otherwise}. \end{aligned}$$

Fig. 1 illustrates an assignment list for the planes. The vertical axis represents the gates available and the horizontal axis is the time. The list is given for a whole day. The planes, depicted as horizontal bars, are shown to occupy the corresponding gates for certain sojourn.

Fig. 2 illustrates some focused area of Fig. 1; that is, the assigned planes to the first 5 gates for the time interval of 8 am to 10 am. In this specific interval, first gate has no assigned planes; whereas, second gate resides the plane 01 for one time slot and the plane 18 for three time slots. Time axis is displayed in discrete version where each day has been divided into $(24 * 60/n)$ timeslots.

3. Big Bang–Big Crunch method

Randomness can be seen as equivalent to the energy dissipation in nature while convergence to a local or global optimum point can be viewed as gravitational attraction. Since energy dissipation creates

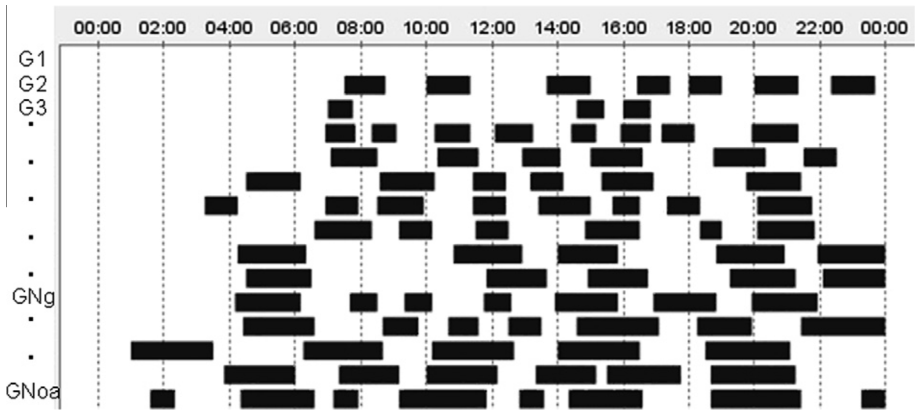


Fig. 1. An arbitrary sample assignment list.

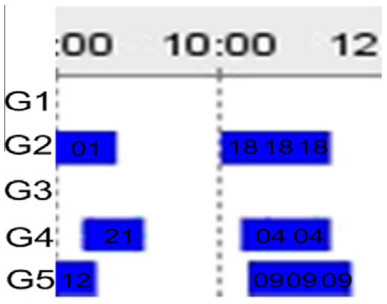


Fig. 2. A focused look into the time slice in between 08:00 and 10:00.

disorder from ordered particles, randomness can be used as a transformation from a converged solution (order) to the birth of totally new solution candidates (disorder or chaos) (Erol & Eksin, 2006).

BB–BC is a population based method as genetic algorithms (GA). Initial population is created randomly. In this phase, the candidate solutions are spread all over the search space in a uniform manner. The initial population creation phase is then followed by the Big Crunch phase. The Big Crunch is a convergence operator that has many inputs but only one output, which can be named as the 'centre of mass', since the only output has been derived by calculating the centre of mass. For a minimization problem, centre of mass can be calculated by weighting the individuals with corresponding fitness evaluations as in (2):

$$\vec{x^c} = \frac{\sum_{i=1}^{N} \frac{1}{f^i}\vec{x^i}}{\sum_{i=1}^{N} \frac{1}{f^i}} \tag{2}$$

In (2), $\vec{x^i}$ is the position vector for the ith individual and f^i stands for the fitness value of the ith individual. Instead of using (2), the fittest individual can be selected as the centre of mass. In this study, the centre of mass selection is preferred.

After the calculation of the centre of mass, the creation of a new population which is called the Bing Bang phase is carried out by calculating new points around the centre of mass using normal distribution function. The designer should handle the impermissible candidates which sprawl beyond the search space at this phase. This can be formalized as in (3):

$$x^{\text{new}} = x^c + lr/k \tag{3}$$

In (3), x^c is the centre of mass, l is the upper limit of the parameter, r is a normal random number and k is the iteration step. Then, the new point is upper and lower bounded. The algorithm continues until predefined stopping criterion has been met. For further details on the Big Bang Big Crunch optimization algorithm one can refer to Erol and Eksin (2006).

4. Heuristic and optimization based solution approaches

In this chapter, firstly a greedy method from the literature will be introduced. Secondly, a new heuristic method that has been named as ground time duration maximization algorithm (GTMA) will be discussed. Finally, the main contribution of this study, implementation of the Single Leap Big Bang–Big Crunch (SL-BBBC) method will be given.

The design of an efficient heuristic becomes a paramount importance and constitutes the key focus of this important application (Xu & Bailey, 2001). Deterministic solutions provide a good initial starting point for the stochastic algorithm. Starting with the best heuristic solution, the stochastic approaches can improve the solution by modifying the assignment list that is given in Fig. 1. On contrary to all the previous work so far done in this area, our new method does not work on the final assignment list, but on the plane ordering process. Plane ordering process can be defined as assigning priority for all the planes with respect to a chosen criterion. Once all the planes are ordered, they are tried to be allocated starting from the one having highest priority. By doing so, all the constraint satisfaction checks needed after a modification on the assignment list can be omitted. Besides this, new approach can be used with any heuristic that constitutes a basis for ordering – or priority assignment – for the flights and with the allocation module in any airport having different constraints. The following subchapters further enlighten the steps of the previous and the suggested approaches and benefits of the new algorithm.

4.1. Heuristic approaches

4.1.1. Previously reported heuristic: Greedy algorithm for minimizing the number of flights assigned to the apron (open air parking place)

In their previous works, Ding et al. (2004a, 2004b, 2005) proposed a greedy algorithm to minimize the number of the un-gated flights. The flights are ordered with respect to departure times and assigned to the gates one by one respecting this order. If there are no gates available, then that flight is assigned to the apron. The algorithm steps are summarized for quick referencing as below:

(1) Sort the flights according to the departure time $T_{D(i)}$.
(2) Set $g_k = -1$ for all gates where g_k $(1 < k < Ng)$ represents the earliest available time in gate-k (that is the departure time of the last assigned plane to gate-k).
(3) For each flight i find gate-k such that $g_k < T_{A(i)}$ and g_k is maximized.
  (i) if such k exists, assign flight i to gate-k, update $g_k = T_{D(i)}$.
  (ii) else assign flight i to the apron.
(4) Output the result.

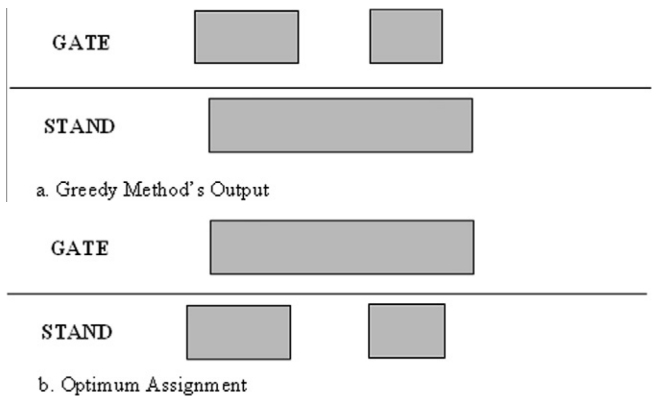


Fig. 3. Illustration for a case of failure of greedy algorithm.

The algorithm is proven to be optimum for minimizing the number of flights assigned to the apron but it has a weakness in maximizing the total gate time if the early departing flight is a short staying one as illustrated in Fig. 3.

4.1.2. A new heuristic approach: ground time duration maximization algorithm (GTMA)

GTMA is designed with the objective to maximize the total gate duration. The underlying idea is to sort planes with respect to their staying durations and then allocating them one by one. That is to say, the longest staying plane is assigned with the highest priority:

(1) Pick the flight with longest time interval between its arrival and departure.
(2) Start from gate #1.
(3) Assign the flight to the gate if possible; else, select the next gate and repeat the procedure until finding a vacant gate.
(4) Remove the flight from the list once it is assigned.
(5) Go to step #1 until all the flights have been assigned.

This heuristic method generates an order for the allocation process as the greedy method. The long staying flights are assigned in the first place and the flight with smaller gate durations can be inserted in between these larger gate durations. However, this method may not be optimal for certain cases with respect to gate time maximization criterion as illustrated in Fig. 4.

4.2. Single Leap Big Bang Big Crunch algorithm (SL-BBBC)

All the studies in GAP that are based on evolutionary algorithms focus on modifying the final assignment list given in Fig. 1 in different ways. This makes the running procedure highly nonlinear and that causes very long run time for the algorithm. This is unfavorable or unacceptable in most cases because frequently occurring delays in the flights pin down a quick reconfiguration of the gate assignment list.

Single Leap Big Bang–Big Crunch (SL-BBBC) algorithm makes its progress on an individual which is initially assigned by the deterministic solution developed by any heuristic plane ordering algorithm. In this study, we will run SL-BBBC algorithm after the heuristic GTMA since it provides much better results compared to greedy heuristic method reported in Section 4.1. In Single Leap Big Bang–Big Crunch (SL-BBBC) algorithm, there is no population of solutions, so no information exchange between solution candidates will take place. For this reason, the BBBC algorithm has been renamed as "Single Leap Big Bang–Big Crunch". The unique solution at hand is modified at each iteration step and if a better solution is attained, then the next iteration works on newly generated solution. In summary, aforementioned GTMA algorithm is used to find deterministic solution and then the solutions are further improved by using SL-BBBC, that is to say, the deterministic algorithms serve initial point for the evolutionary algorithm.

In our study, the key point is that the SL-BBBC algorithm works on the assignment order of planes instead of the final assignment list itself. Once the initial assignment list and the order of plane assignment have been obtained, SL-BBBC algorithm is conducted on the assignment order for further improvement.

The flight list is input for the plane ordering module, which is followed by the allocation module. The final output is the assignment list (Fig. 5).

We can summarize the new assignment methodology as follows:

(1) Apply GTMA and find an assignment list.
(2) Log the order in which the planes are assigned.
(3) Apply SL-BBBC to find better assignment order.

SL-BBBC algorithm is implemented in three different ways:

(a) Interchanging the order of only two flights with random distances away from a random center in the list. In Fig. 6, the center is chosen to be the position of *Flight #3* and the distance is chosen to be two units. Then *Flight #1* and *Flight #5* interchange the positions. Note that the "distance" here

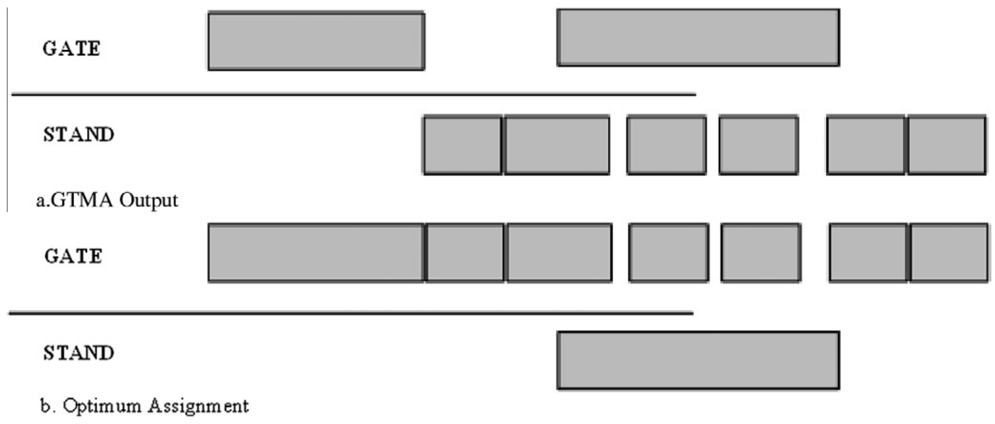
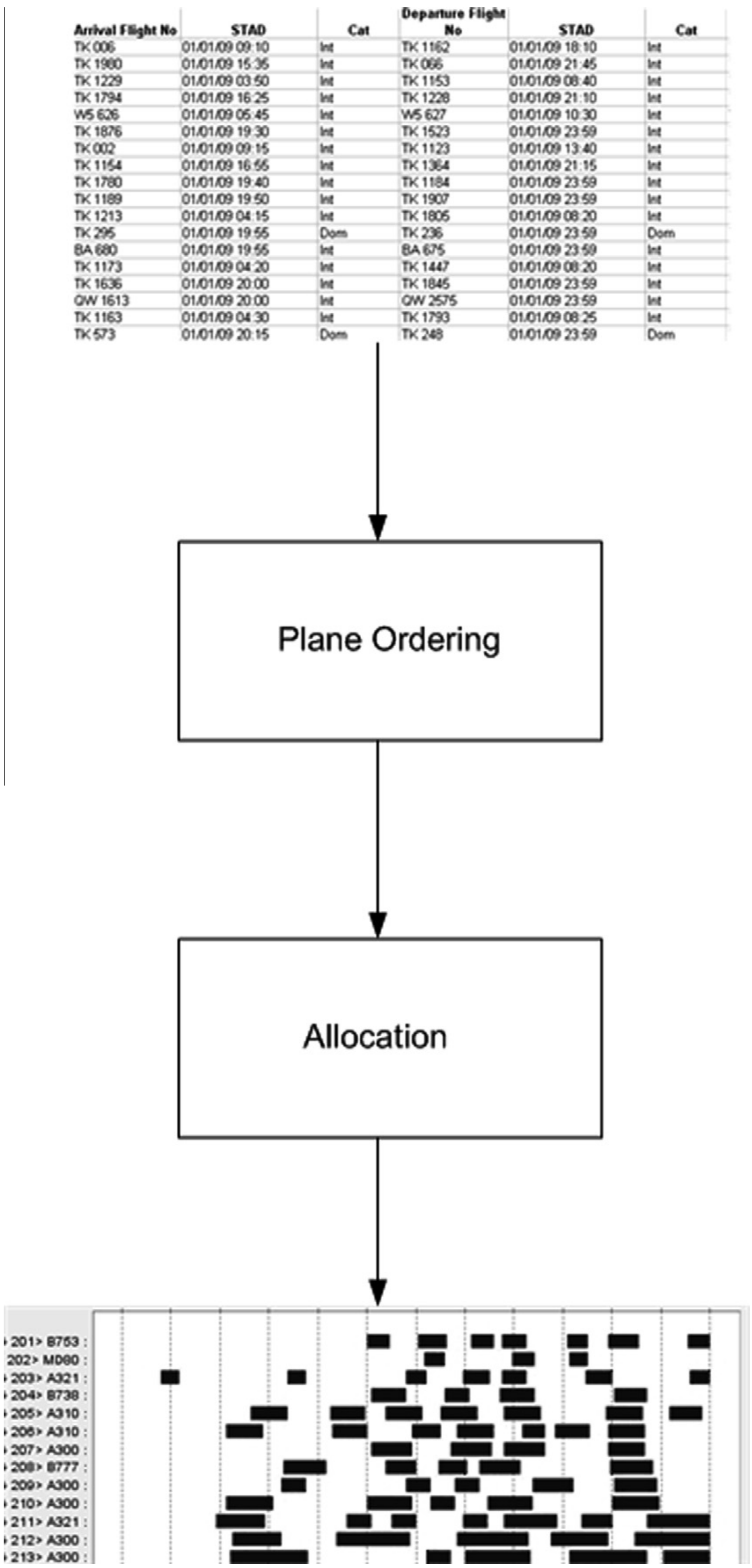


Fig. 4. Illustration for a case of failure of GTMA.



Arrival Flight No	STAD	Cat	Departure Flight No	STAD	Cat
TK 006	01/01/09 09:10	Int	TK 1162	01/01/09 18:10	Int
TK 1980	01/01/09 15:36	Int	TK 066	01/01/09 21:45	Int
TK 1229	01/01/09 03:50	Int	TK 1153	01/01/09 08:40	Int
TK 1794	01/01/09 16:25	Int	TK 1228	01/01/09 21:10	Int
W5 626	01/01/09 05:45	Int	W5 627	01/01/09 10:30	Int
TK 1876	01/01/09 19:30	Int	TK 1523	01/01/09 23:59	Int
TK 002	01/01/09 09:15	Int	TK 1123	01/01/09 13:40	Int
TK 1154	01/01/09 16:55	Int	TK 1364	01/01/09 21:15	Int
TK 1780	01/01/09 19:40	Int	TK 1184	01/01/09 23:59	Int
TK 1189	01/01/09 19:50	Int	TK 1907	01/01/09 23:59	Int
TK 1213	01/01/09 04:15	Int	TK 1806	01/01/09 08:20	Int
TK 295	01/01/09 19:55	Dom	TK 236	01/01/09 23:59	Dom
BA 680	01/01/09 19:55	Int	BA 675	01/01/09 23:59	Int
TK 1173	01/01/09 04:20	Int	TK 1447	01/01/09 08:20	Int
TK 1636	01/01/09 20:00	Int	TK 1845	01/01/09 23:59	Int
QW 1613	01/01/09 20:00	Int	QW 2575	01/01/09 23:59	Int
TK 1163	01/01/09 04:30	Int	TK 1793	01/01/09 08:25	Int
TK 573	01/01/09 20:15	Dom	TK 248	01/01/09 23:59	Dom

Fig. 5. Basic flow diagram of the allocation process.

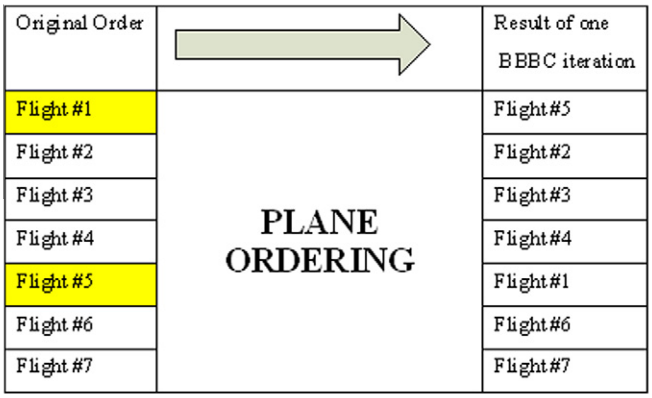


Fig. 6. Illustration of SL-BBBC version-a.

is related with the "explosion strength" and "center" is related with the "center of mass" in the original BBBC algorithm (Erol & Eksin, 2006).

(b) Randomly permuting the flights in between randomly selected two points. In Fig. 7, all the flights between *Flight #2* and *Flight #6* are reordered. Number of flights to be reordered is correlated with the explosion strength in original BBBC algorithm.

(c) Interchanging the order of N random flight pairs with random distances away from random centers in the list. This is a generalized version for case (a). Here, both the number of changes and the distances in between are related with the explosion strength that is getting smaller as the number of iterations increase.

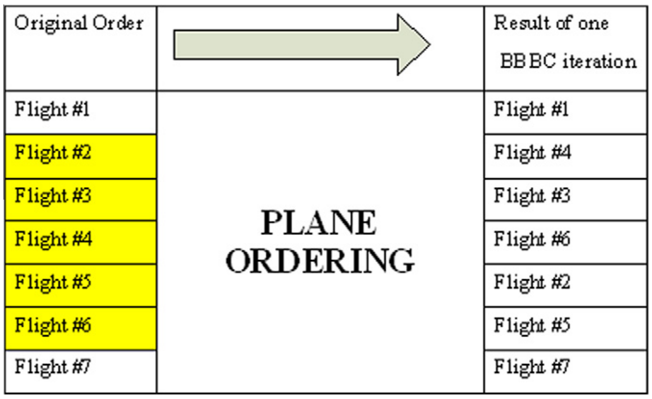


Fig. 7. Illustration of SL-BBBC version-b.

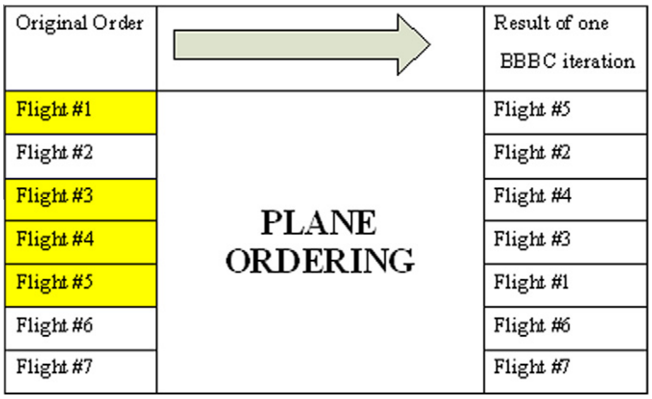


Fig. 8. Illustration of SL-BBBC version-c.

Fig. 8 illustrates the interchanging scheme where $N = 2$. Here, the interchange between flights numbered 1, 5 and the flights numbered 3, 4 is illustrated as an example.

5. Simulation results

5.1. Simulation results with artificially generated data set

In this section, the performance results are provided so as to demonstrate the effect of SL-BBBC method over test data sets. We have developed our own test data generator which takes the following parameters into consideration as inputs:

(1) The proportion in between total time slot demand and total available discrete time slots, d.
(2) Prime time traffic factors, p_1 and p_2.
(3) Mean staying time for a plane, m.
(4) Standard deviation for staying times of all arranged flights, σ.

The user of this test data generator can define at most two "prime time"; that is to say, high gate demand time interval during a day. When a flight is generated by the adjusted parameters, it will be assigned to each prime time with a probability equal to the chosen factor of the corresponding prime time. Even if it is not assigned to prime time by this step, the flight can still be assigned to that region by coincidence. The test data generator makes up the whole flight list accordingly. The mean, standard deviation and prime time factors are directly used in daily flight list generation. However, the ratio of demanded slot/available slot (=demanded gate duration/available gate time) is used to find the necessary plane number and then this value is assigned with some uniform random number in the vicinity of 10%. This simple manipulation is done just for the diversity of the plane numbers for batch data file generation. Fig. 9 shows the graphical user interface for the test data generation software. For a sample (not optimized) view for the selected parameters, one may use *Sample allocation* button. Then, *Generate* button produces data files derived by the selected parameters at the instant. Data generation steps for a single day can be given as follows:

(1) Find total gate duration demanded, Tt, in terms of discretized time slot number:

$$Tt = d * Ng * Nt \tag{4}$$

where Ng is the number of gates and Nt is the number of time slots in a whole day.

(2) Find number of planes, N, to be generated,

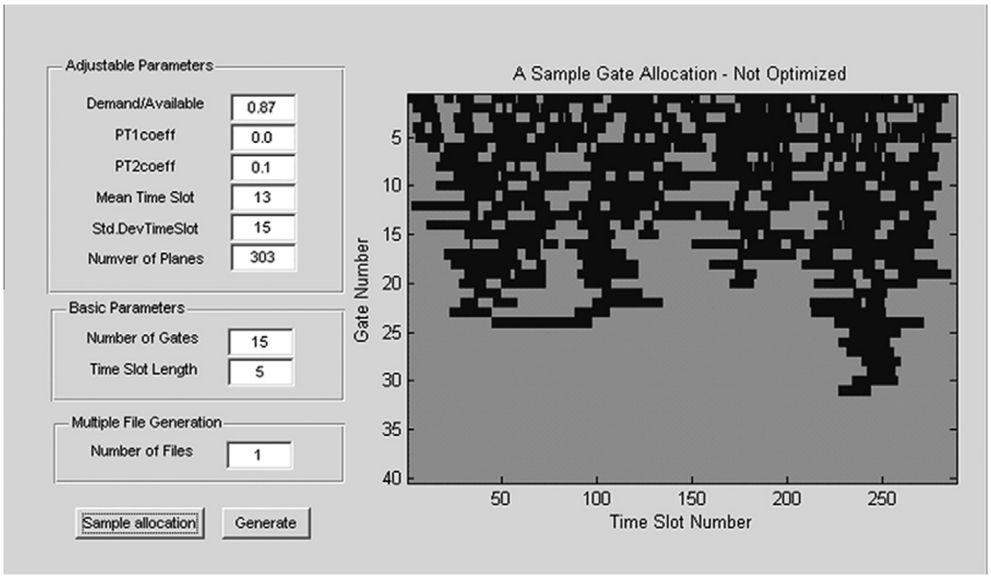
$$N = \mathrm{round}(Tt/m + 10 * rand) \tag{5}$$

where *rand* is a uniform random number in the interval $[-1, 1]$; *round* function produces the nearest integer as the number of planes should be an integer value.

(3) For all N planes, pick up an integer gate duration value from normal distribution with mean m and standard deviation σ that are defined by the user.
(4) For all N planes, assign the plane to the corresponding prime time region with a probability chosen by the user.
(5) For the planes not allocated to the prime time regions, randomly assign arrival indexes that are convenient with the gate time determined in step 3.

The number of files to be produced can also be changed. *Number of gates* parameter is arranged to depict preferable first Ng gates. In this specific example, they are selected as follows:

$Ng = 15$, $Noa = 25$ and $Ns = 40$.


Adjustable Parameters

Demand/Available 0.87

PT1coeff 0.0

PT2coeff 0.1

Mean Time Slot 13

Std.DevTimeSlot 15

Numver of Planes 303

Basic Parameters

Number of Gates 15

Time Slot Length 5

Multiple File Generation

Number of Files 1

Fig. 9. The view of the test data generator user interface.

We have generated three different files representing three different characteristics for a flight schedule.

(1) *Moderate data set:* this data set structure is close to data set structures observed in Turkey Airports. There is a relatively high demand for certain time slots during the morning and evening. Average gate time is close to an hour and median value is nearly half an hour. Demanded gate time does not exceed available gates. That is to say, the ratio of demanded time slot/available time slot is less than one. But for sure, there occurs un-gated flights due to lack of perfect fitting of gate durations.
(2) *High gate demand distributed uniformly:* there are considerably larger demand for the same number of gates with data set-1. The ratio of demanded time slot/available time slot is slightly larger than one and this causes irrepressible un-gated flights. There are no intended peaks on the gate demands throughout the day.
(3) *High gate demand with demand peaks:* demanded gate duration/available gate time is same with data set-2. In this data set, the mean staying time for the flights are quite decreased (represents the flight schedule for an airport having very crowded traffic and many connection flights) and two peak demand regions are defined, one of them hosting the 35% of the total flights.

Every experiment is carried for 100 times to produce reliable statistics since the running algorithms have stochastic nature. The results are analyzed to yield mean, median, standard deviation, maximum and maximum of deviations and some of these are reported whenever appropriate.

Each stand at the airport has full vacancy at start. The whole day is divided into 5 min time intervals summing up $24 * 60/5 = 288$ time slots. When scoring an assignment list, first $Ng = 15$ stands are concerned. Each assigned time slot at the first Ng stands equally contributes to the scoring. For example, a plane arriving at 08:00 am and departing at 11:00 am stays for 36 time slots and if the plane can be assigned to one of the score contributing stands, the overall score for the assignment list will increase by 36.

Table 1 reports the mean value results over 30-days in a compact view. Note that the SL-BBBC algorithm is only allowed to work for 2500 fitness evaluations taking less than 1 min in Intel Core 2 Duo Processor. Though three different approaches for SL-BBBC implementation have been tried through simulations and experiments, only the last one coded as SL-BBBC version-c is reported since it yielded the most successful results. Figs. 10–12 show the cost scores of the three algorithms with respect to days.

Table 1
Mean cost values for synthetic dataset (each consists of 30 days data).

Method	Moderate data set	High gate demand distributed uniformly	High sate demand having demand peaks
Greedy method	2847.23	3004.50	291377
GTMA	3440 33	3715.56	3285.63
SL-BBBC	3483.67	3760.08	3308.20

5.2. Simulation results with actual field data

In this part, the experiments are performed on the data collected from the Atatürk Airport in İstanbul. Data collected are for 31 days of month January, 2009 and represent an average of 300

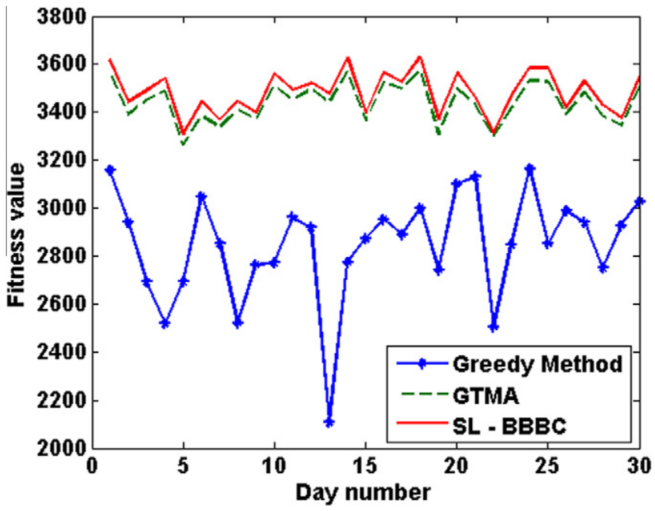


Fig. 10. Comparison of the three algorithms for moderate data set.

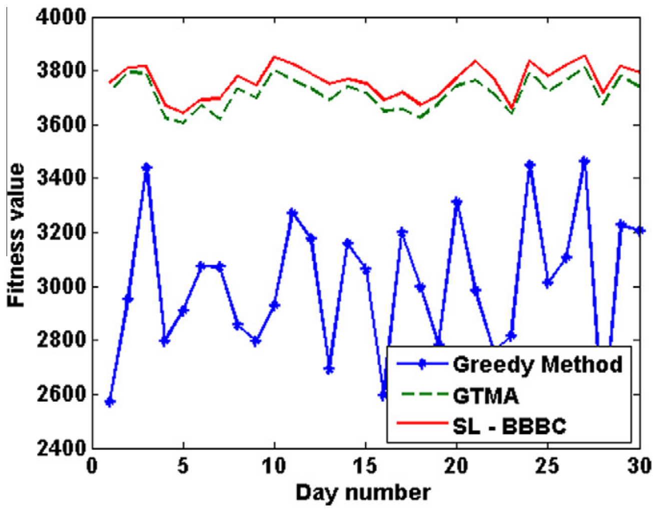


Fig. 11. Comparison of the three algorithms for high gate demand distributed uniformly.

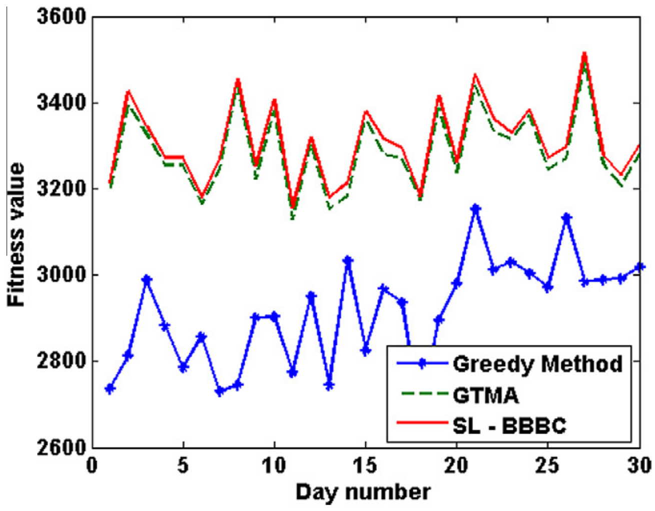


Fig. 12. Comparison of the three algorithms for high gate demand with demand peaks.

planes per day. Although the gate and plane constraints can be handled for this specific airport, as mentioned earlier, the problem at hand can be set as squeezing maximum planes to the gates.

All the testing procedure and analyzing parameters are the same with tests in part (a) of this last section. Table 2 reports the cost values of *random ordering*, in which the planes are ordered randomly (average on 100 random ordering for each day results are averaged over whole days); *greedy method*, given in Section 4.1; *GTMA*, given in Section 4.2; *random re-ordering* over GTMA, where heuristic method's outputs are randomly interchanged for the same iteration number as SL-BBBC; and finally *SL-BBBC* method.

Table 2
Mean cost value for 31 days.

Method	Real world data set
Random ordering	2866.71
Greedy method	2761.52
GTMA	3327.39
Random re-ordering	3333.00
SL-BBBC	3371.53

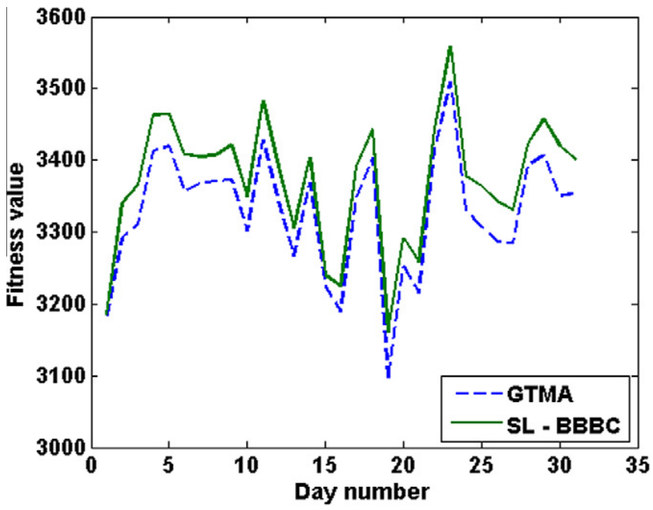


Fig. 13. Comparison of GTMA and SL-BBBC in a real world data set.

Greedy method performs even worse than random ordering average in this data set. Besides, if one starts from a good initial point then it is observed that interchanging the plane orders in a totally random manner makes not much difference in performance; whereas, being a systematic method, applying SL-BBBC algorithm for such good initial conditions is still able to create respectful difference in performance. Note that the average cost score of random ordered allocations is somewhere near 2866 and GTMA heuristic improves this score by 16%. The stochastic neighborhood search method further improves the results by 9.6% of the previous improvement. That is to say SL-BBBC algorithm starts from a quite acceptable solution and further improves the solution; on the other hand, if it had been started from a random solution candidate (that is a random ordering of the planes) the improvement would have been much more in the expense of process time. Hybridizing the algorithms by initial solution generation by a heuristic optimizes the solution both in terms of objective function value and process time. Considering the annual profit obtained by this final improvement, one can easily justify the importance of the new algorithms. Fig. 13 clearly shows the improvement gained in using the SL-BBBC method in daily basis.

Since the total run time for the SL-BBBC algorithm is less than one minute it allows quick restructuring of the assignment table. That is one of the most powerful aspects of the algorithm for the practical applications. The method is compatible with any cost function evaluation but the algorithm speed heavily depends on cost function process time. Moreover, if the algorithm were allowed to evaluate more candidate fitness values, the objective function value scores could have been further improved.

6. Application at Atatürk Airport of İstanbul

The proposed algorithm is used on a real world application to work both as an off-line and on-line gate allocation module in one of the most frequented airports of Europe, İstanbul Atatürk Airport. The software developed is a resource management system having the architecture given in Fig. 14. The gate assignment automation is implemented on ROTA Engine Server. The detailed explanations for the other components are beyond the scope of this work and deliberately omitted here.

The resource management system (RMS) can be used as a web-based or desktop application. Thus, a variety of users with different devices throughout the airport can utilize the system in a collaborative manner. The user interface, *Dashboard*, includes touch screen

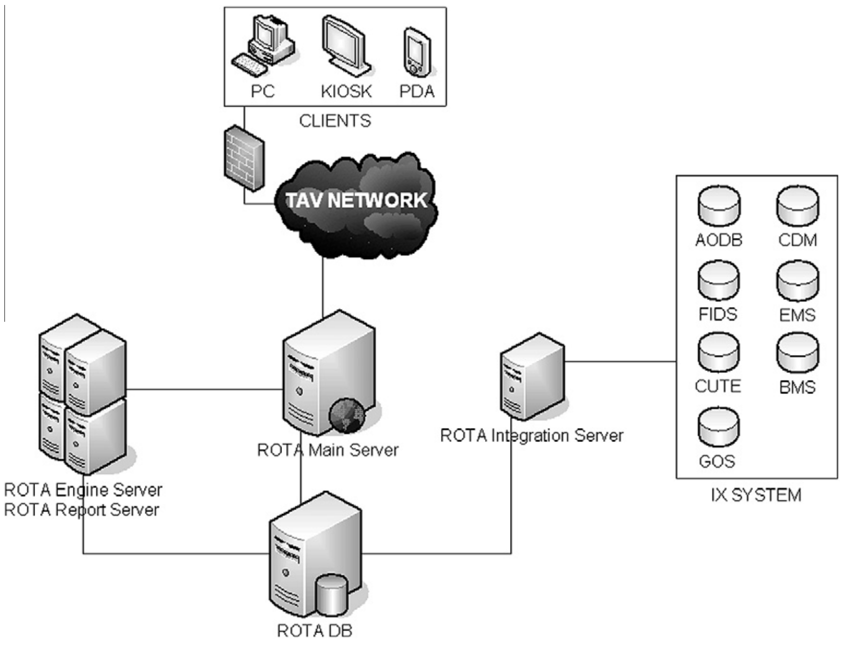
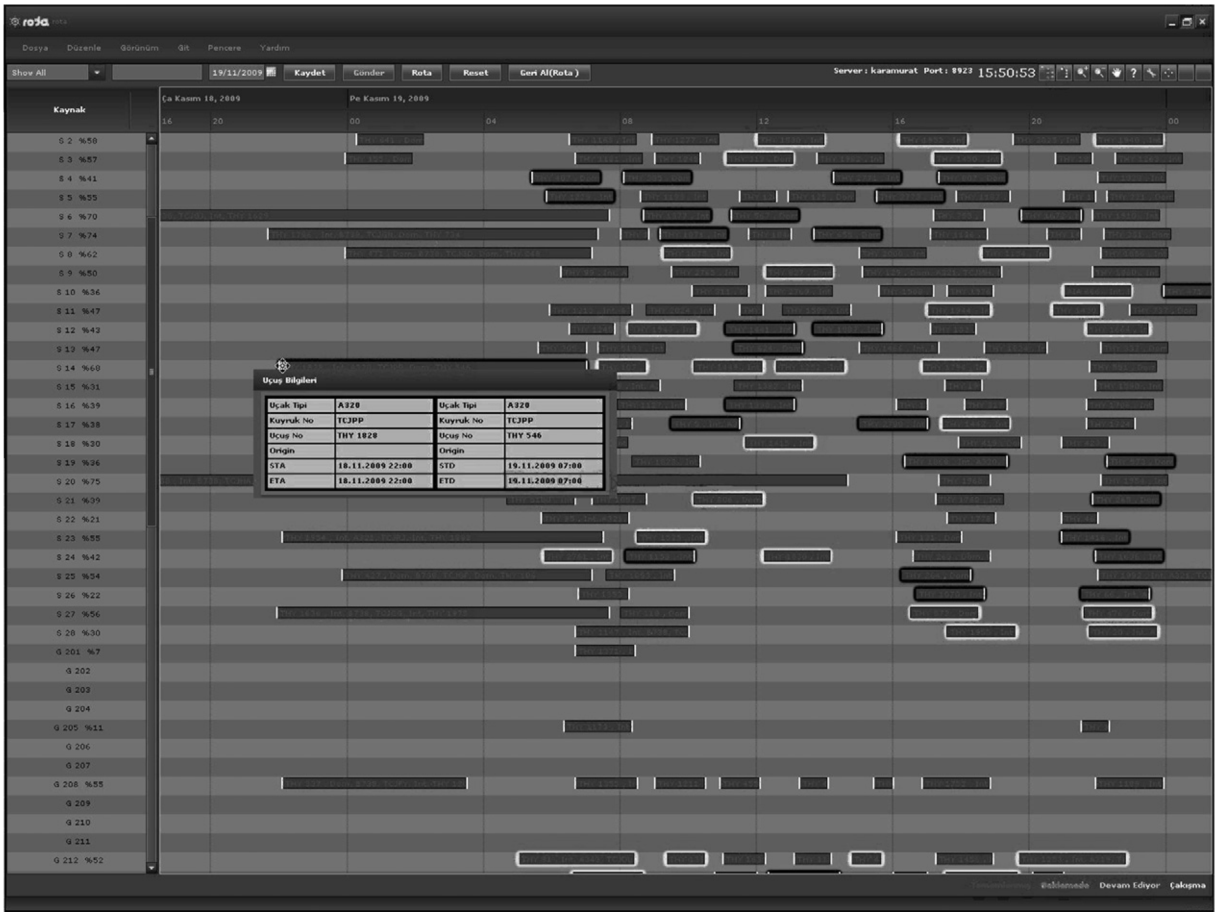


Fig. 14. The architecture for the resource management system.

capability to maximize usability and control. *Dashboard* is designed mainly for maximizing monitoring capabilities according to the needs of control centre staff. Figs. 15–17 gives some example screenshots from the *Dashboard*.

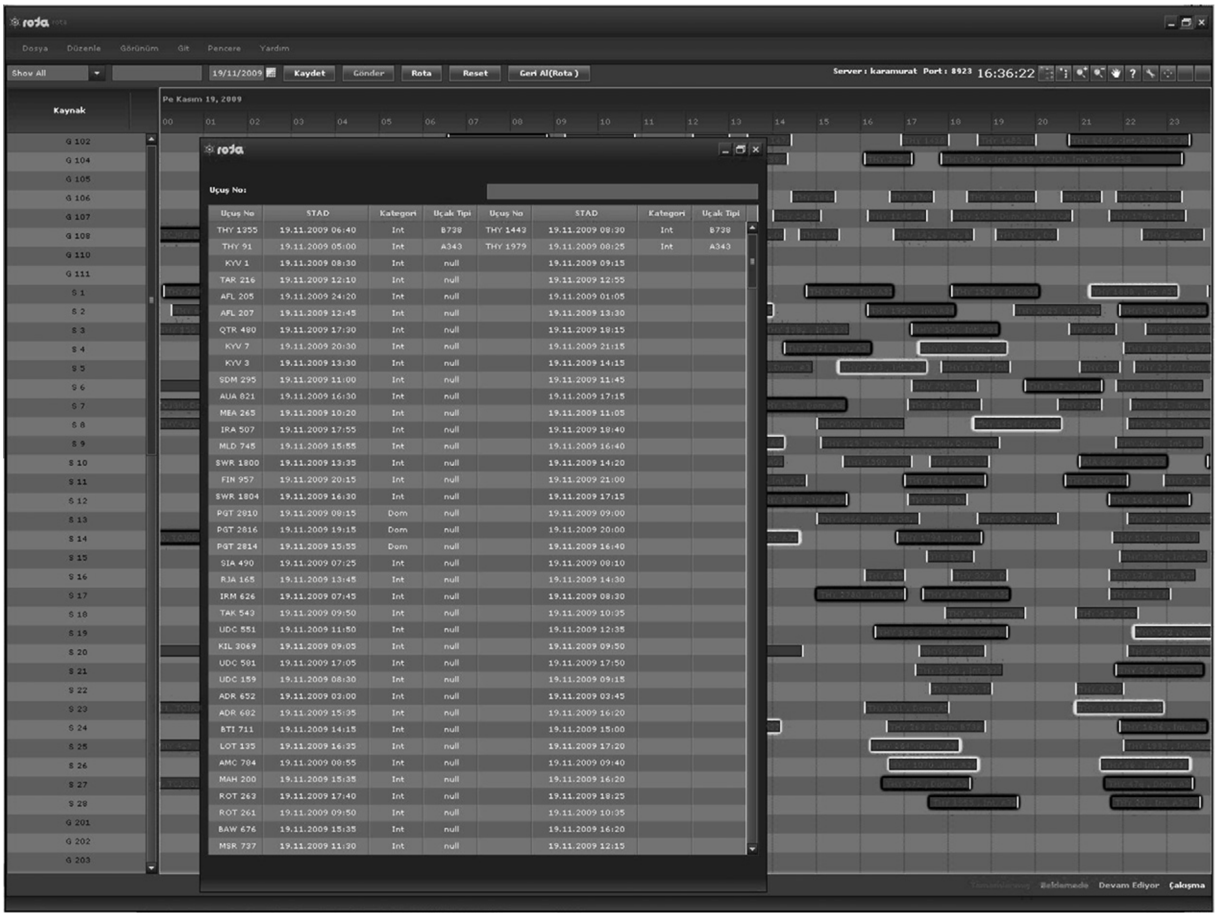
The operator can display the statistics of assignment as well as the basic information about a particular flight. The flight list and the assignment list; that is the inputs and outputs of the system are displayed concurrently.



Uçuş Bilgileri

Uçak Tipi	A320	Uçak Tipi	A320
Kuyruk No	TCJPP	Kuyruk No	TCJPP
Uçuş No	THY 1828	Uçuş No	THY 546
Origin		Origin	
STA	18.11.2009 22:00	STD	19.11.2009 07:00
ETA	18.11.2009 22:00	ETD	19.11.2009 07:00

Fig. 15. An example gate allocation screen when flight information window is open.



Uçuş No	STAD	Kategori	Uçak Tipi	Uçuş No	STAD	Kategori	Uçak Tipi
THY 1355	19.11.2009 06:40	Int	B738	THY 1443	19.11.2009 08:30	Int	B738
THY 91	19.11.2009 05:00	Int	A343	THY 1979	19.11.2009 08:25	Int	A343
KYV 1	19.11.2009 08:30	Int	null		19.11.2009 09:15		
TAR 216	19.11.2009 12:10	Int	null		19.11.2009 12:55		
AFL 205	19.11.2009 24:20	Int	null		19.11.2009 01:05		
AFL 207	19.11.2009 12:45	Int	null		19.11.2009 13:30		
QTR 480	19.11.2009 17:30	Int	null		19.11.2009 18:15		
KYV 7	19.11.2009 20:30	Int	null		19.11.2009 21:15		
KYV 3	19.11.2009 13:30	Int	null		19.11.2009 14:15		
SDM 295	19.11.2009 11:00	Int	null		19.11.2009 11:45		
AUA 821	19.11.2009 16:30	Int	null		19.11.2009 17:15		
MEA 265	19.11.2009 10:20	Int	null		19.11.2009 11:05		
IRA 507	19.11.2009 17:55	Int	null		19.11.2009 18:40		
MLD 745	19.11.2009 15:55	Int	null		19.11.2009 16:40		
SWR 1800	19.11.2009 13:35	Int	null		19.11.2009 14:20		
FIN 957	19.11.2009 20:15	Int	null		19.11.2009 21:00		
SWR 1804	19.11.2009 16:30	Int	null		19.11.2009 17:15		
PGT 2810	19.11.2009 08:15	Dom	null		19.11.2009 09:00		
PGT 2816	19.11.2009 19:15	Dom	null		19.11.2009 20:00		
PGT 2814	19.11.2009 15:55	Dom	null		19.11.2009 16:40		
SIA 490	19.11.2009 07:25	Int	null		19.11.2009 08:10		
RJA 165	19.11.2009 13:45	Int	null		19.11.2009 14:30		
IRM 626	19.11.2009 07:45	Int	null		19.11.2009 08:30		
TAK 543	19.11.2009 09:50	Int	null		19.11.2009 10:35		
UDC 551	19.11.2009 11:50	Int	null		19.11.2009 12:35		
KIL 3069	19.11.2009 09:05	Int	null		19.11.2009 09:50		
UDC 581	19.11.2009 17:05	Int	null		19.11.2009 17:50		
UDC 159	19.11.2009 08:30	Int	null		19.11.2009 09:15		
ADR 652	19.11.2009 03:00	Int	null		19.11.2009 03:45		
ADR 682	19.11.2009 15:35	Int	null		19.11.2009 16:20		
BTI 711	19.11.2009 14:15	Int	null		19.11.2009 15:00		
LOT 135	19.11.2009 16:35	Int	null		19.11.2009 17:20		
AMC 784	19.11.2009 08:55	Int	null		19.11.2009 09:40		
MAH 200	19.11.2009 15:35	Int	null		19.11.2009 16:20		
ROT 263	19.11.2009 17:40	Int	null		19.11.2009 18:25		
ROT 261	19.11.2009 09:50	Int	null		19.11.2009 10:35		
BAW 676	19.11.2009 15:35	Int	null		19.11.2009 16:20		
MSR 737	19.11.2009 11:30	Int	null		19.11.2009 12:15		

Fig. 16. An example gate allocation screen when manual edit window is open.

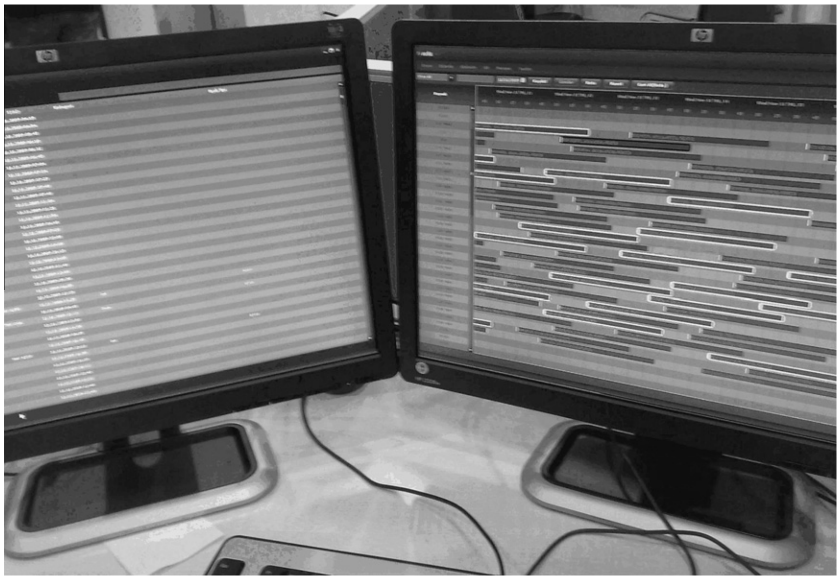


Fig. 17. Operator display for gate allocating (taken from Atatürk Airport with courtesy of TAV Bilişim A.Ş.).

The core module of RMS is the rule and optimization engine. The rule engine enables authorized personnel to model the constraints and relationships both for resources and tasks through all their possible attributes according to market considerations and airline/agent preferences. These task-oriented rules can be based on the following groups and extended according to other classes and attributes:

(i) Airport-based.
(ii) Terminal-based.

(iii) Airline-based.
(iv) Based on registration number.
(v) Other (i.e. recurring assignment rules, etc.).

Predictably, because planning staff will utilize so many of the above constraints, it is inevitable that some of them will overlap. Although, in some cases, this can be a preferred result in terms of operational workload, some unexpected and undesired results may occur under normal circumstances. The rule engine module resolves this issue with a scoring mechanism and the overlapped constraints can be managed using this functionality. In addition to scoring and constraints, there is another functional parameter that can be accessed in modeling constraints called *soft constraints (preferences)*. These constraints refer to the rules which can be violated by the optimization engine under some circumstances in order to meet functional objectives. At any given time during an operation (and also in planning), if there is a shortage of resources, the system proactively generates automatic conflict messages with pre-defined solutions. It would be appropriate to point out at this stage that the SL-BBBC optimization algorithm works in an independent manner from constraint generation.

Another feature in RMS is the optimization engine. This submodule provides optimization of daily tasks based on pre-defined flight lists. The main idea in developing the optimization engine is to serve the priorities set by airport operators according to their management policies and preferences. Some of the common objective functions are given below:

(a) Revenue maximization through the optimization of gate and stand assignments.
(b) Maximizing the utilization of gate capacity.
(c) Maximizing airport capacity.
(d) Enhancing overall service quality (punctual departures, cost competitiveness and reliability).
(e) Minimizing the walking distance for departing and arriving passengers to provide smooth passenger flow.

It should be noted that the above mentioned objectives can be evaluated together according to their priority levels in an approach known as multi-objective optimization. The cost function is evaluated as a weighted sum of the all listed above. It will again be appropriate to underline that SL-BBBC algorithm used here does not need to know this fitness function; but just needs the assigned score to a specific candidate solution to proceed.

The performance results for the resource management system are given in Table 3 and Fig. 18. In Table 3, the data collected for the month February of 2010 (28 days) are used and the scores are reported as the average for the month. The GTMA heuristic improves the cost by 12.49% with respect to manual (random) allocations and applying SL-BBBC algorithm for 100 iterations (=100 "leaps") improves the results by 22.68% of the previous improvement. For 1000 iterations this improvement value is 34.59%. Fig. 18 gives the results in daily basis. We have omitted the results for SL-BBBC algorithm for 100 iterations in order to decrease figure complexity.

Table 3
Mean cost values for the data collected from resource management system of Atatürk Airport for month February 2010.

Method	Cost score
Manual allocations	42.34
GTMA	47.63
SL-BBBC (100 evaluations)	48.83
SL-BBBC (1000 evaluations)	49.46

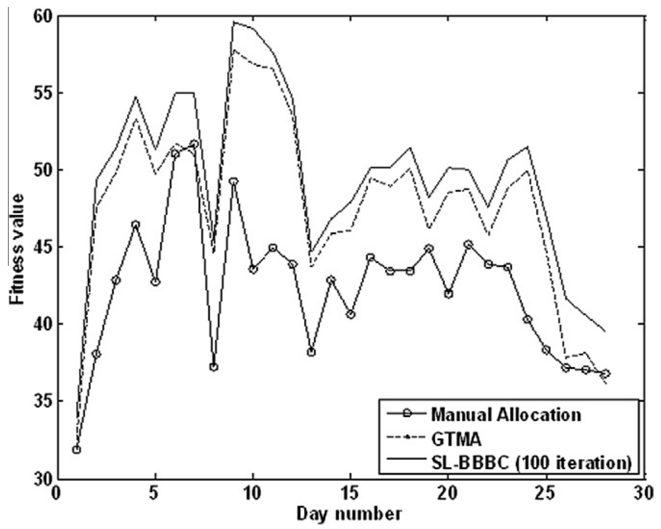


Fig. 18. Comparison of the algorithms running on Atatürk Airport. Data are the 28 days of February, 2010.

7. Conclusion

In this paper, the airport gate assignment problem is considered so as to maximize the total gate duration of the flights assigned to the gates. Then, the airport gate assignment problem turns out to be maximizing the total sojourn in the first *Ng* gates. Here, a new stochastic approach has been introduced to the problem utilizing a problem specific modification of Big Bang–Big Crunch optimization algorithm, namely Single Leap Big Bang Big Crunch (SL-BBBC). The key feature of this problem based modified evolutionary type of optimization algorithm, that is also the one of the main contributions of this work, is to interchange the queue order of the planes (= flights) to be assigned rather than interchanging the positions of the *N* planes that are already assigned. That is to say, the algorithm steps do not interact with the assignment strategy and they just exchange the order of plane handling by the determined strategy. This modularity of SL-BBBC makes it compatible with any assignment logic. We showed this hybridized approach on a simple yet effective heuristic algorithm which we abbreviated as GTMA. Starting from a good initial solution obtained by this heuristic and then using our stochastic method is rather effective in terms of process time and the method proposed can be used in all practical applications.

The results obtained through simulation examples and experiments with real world data show the effectiveness of the allocation strategy. Besides, the modularity of the plane ordering logic provides us great flexibility to work with any constraint processing engine. Moreover, this new algorithm does not require any objective score calculation and does not have to know details on constraints or cost calculation. These facts are tried to be illustrated at the final section dedicated to the application study done on the biggest airport of Turkey.

References

Babic, O., Teodorovic, D., & Tosic, V. (1984). Aircraft stand assignment to minimize walking. *Journal of Transportation Engineering, 110*(1), 55–66.
Bandara, S., & Wirasinghe, S. (1992). Walking distance minimization for airport terminal configurations. *Transportation Research, 26A*, 59–74.
Bolat, A. (1999). Assigning arriving flights at an airport to the available gates. *Journal of the Operational Research Society, 50*, 23–34.
Bolat, A. (2001). Models and a genetic algorithm for static aircraftgate assignment problem. *Journal of the Operational Research Society, 52*, 1107–1120.
Chang, C. (1994). *Flight sequencing and gate assignment in airport hubs*. Ph.D. Thesis. University of Maryland at College Park.

Ding, H., Lim, A., Rodrigues, B., & Zhu, Y. (2004a). Aircraft and gate scheduling optimization at airports. In *Proceedings of the 37th Hawaii IEEE international conference on system sciences* (pp. 1–8). ISBN: 0-7695-2056-1/04.

Ding, H., Lim, A., Rodrigues, B., & Zhu, Y. (2004b). New heuristics for over-constrained flight to gate assignments. *Journal of the Operational Research Society, 55*, 760–768.

Ding, H., Lim, A., Rodrigues, B., & Zhu, Y. (2005). The over-constrained airport gate assignment problem. *Computers and Operations Research, 32*, 1867–1880.

Dorndorf, U., Drexl, A., Nikulin, Y., & Pesch, E. (2007). Flight gate scheduling: State-of-the-art and recent developments. *The International Journal of Management Science, 35*, 326–334.

Erol, O. K., & Eksin, I. (2006). A new optimization method: Big Bang–Big Crunch. *Advances in Engineering Software, 37*, 106–111.

Gu, Y., & Chung, C. A. (1999). Genetic algorithm approach to aircraft gate reassignment problem. *Journal of Transportation Engineering, 125*(5), 384–389.

Haghani, A., & Chen, M. C. (1998). Optimizing gate assignments at airport terminals. *Transportation Research, 32*(6), 437–454.

Hu, X. B., & Paulo, E. D. (2007). An efficient genetic algorithm with uniform crossover for the multi-objective airport gate assignment problem. In *IEEE congress on evolutionary computation* (pp. 55–62). ISBN: 1-4244-1340-0/07.

Obata, T. (1979). *The quadratic assignment problem: Evaluation of exact and heuristic algorithms*. Technical Report TRS-7901. Troy, New York: Rensselaer Polytechnic Institute.

Teodorovic, D., & Guberinic, S. (1984). Optimal dispatching strategy on an airline network after a schedule perturbation. *European Journal of Operations Research, 15*, 178–182.

Teodorovic, D., & Stojkovic, G. (1990). Model for operational daily airline scheduling. *Transportation Planning and Technology, 14*, 273–285.

Wei, D., & Liu, C. (2007). Optimizing gate assignment at airport based on genetic-Tabu algorithm. In *Proceedings of the IEEE international conference on automation and logistics, Jinan, China* (pp. 1135–1140).

Wirasinghe, S., & Bandara, S. (1990). Airport gate position estimation for minimum total costs – Approximate closed form solution. *Transportation Research, 24B*, 287–297.

Xu, J., & Bailey, G. (2001). The airport assignment problem: Mathematical model and a Tabu search algorithm. In *Proceedings of the 34th Hawaii IEEE international conference on system sciences* (pp. 1–10). ISBN: 0-7695-0981-9/01.