Universitat Autònoma de Barcelona

escola d'enginyeria

Optimització del procés d'embarcament d'aeronaus atenent a les característiques individuals dels passatgers

Treball de fi de grau

Memòria del Treball de Fi de Grau en Gestió Aeronàutica

realitzat per

Gerard Carmona Budesca

i dirigit per

Dr. Angel A. Juan Pérez

Sabadell, a 8 de Juliol de 2013

Lliurament Treball Fi de Grau

Grau en Gestió Aeronàutica

Escola d'Enginyeria (Campus de Sabadell) (UAB)

TÍTOL PROJECTE: Optimització del procés d'embarcament d'aeronaus atenent a les característiques individuals dels passatgers

ALUMNE: Gerard Carmona Budesca

DATA LLIURAMENT: 8 de Juliol de 2013

DIRECTOR: Angel A. Juan Pérez

TITULACIÓ: Grau en Gestió Aeronàutica

MATERIAL LLIURAT

3 Còpies de la MEMÒRIA (signades per l'alumne i el director)

CD-R de la MEMÒRIA

2 Còpies del full d'autorització de dipòsit digital de la memòria

Observacions:

Segell del Centre

Signatura de l'alumne

FULL DE RESUM – TREBALL DE FI DE GRAU DE L'ESCOLA D'ENGINYERIA

Títol del projecte: Optimització del procés d'embarcament d'aeronaus atenent a les característiques individuals dels passatgers	
Autor: Gerard Carmona Budesca	**Data:** 8 de Juliol de 2013
Tutor: Dr. Angel A. Juan Pérez	
Titulació: Grau en Gestió Aeronàutica	

Paraules clau

• **Català:** Estratègies d'embarcament, Optimització, Procés d'embarcament d'aeronaus, Simulació

• **Castellà:** Estrategias de embarque, Optimización, Proceso de embarque de aeronaves, Simulación

• **Anglès:** Aircraft boarding process, Boarding strategies, Optimization, Simulation

Resum del projecte

• **Català:** La optimització del procés d'embarcament de passatgers en aeronaus és una de les vies a través de la qual les companyies aèries poden reduir el temps d'escala als aeroports i maximitzar la utilització dels seus avions. En aquest projecte s'estudia el problema d'embarcament d'aeronaus amb l'objectiu de dissenyar una metodologia que permeti definir una estratègia d'embarcament òptima que resulti en una minimització del temps total d'embarcament tenint en compte les característiques individuals dels passatgers i les relacions existents entre ells.

• **Castellà:** La optimización del proceso de embarque de pasajeros en aeronaves es una de las vías a través de la cual las compañías aéreas pueden reducir el tiempo de escala en los aeropuertos y maximizar la utilización de sus aviones. En este proyecto se estudia el problema de embarque de aeronaves con el objetivo de diseñar una metodología que permita definir una estrategia de embarque óptima que resulte en una minimización del tiempo total de embarque teniendo en cuenta las características individuales de los pasajeros y las relaciones existentes entre ellos.

• **Anglès:** The aircraft boarding process is one of the ways in which airlines can reduce the turnaround time at the airport in order to maximize the utilization rate of their fleet. In this project, the aircraft boarding problem is studied with the aim of designing a boarding strategy that results in a reduction of the total boarding time taking into account the passenger's individual traits as well as the underlying relationships among them.

M'agradaria expressar el més profund agraïment al Dr. Angel A. Juan per la seva ajuda i suport al llarg de tot aquest projecte i al programador Enoc Martinez per haver-me ajudat en el desenvolupament i implementació del programa en JAVA.

Índex

Secció 4: Implementació i validació de l'algorisme

Secció 5: Experiments computacionals

Secció 6: Conclusions i treball futur

Referències i enllaços

Annex

0. Resum

El fort creixement que ha experimentat el transport aeri durant les últimes dècades junt amb la seva liberalització i el conseqüent augment de la competència, ha forçat a les companyies aèries a optimitzar tots els processos sobre els quals tenen control per tal d'augmentar l'eficiència de les seves operacions i assegurar la seva permanència en el mercat. Donat que les aerolínies només generen ingressos quan els seus avions estan volant, el temps que aquests es troben a terra ha de ser minimitzat, fet que implica reduir el temps d'escala als aeroports. L'objecte principal d'aquest estudi, l' optimització del procés d'embarcament de passatgers en aeronaus, és una de les vies a través de les quals és possible reduir la duració total de l'escala. Això s'aconsegueix mitjançant la utilització d'estratègies d'embarcament que permeten definir l'ordre en què els passatgers embarquen l'aeronau.

En l'estudi, en primer lloc, es defineixen les principals estratègies d'embarcament existents i es duu a terme una revisió de la literatura científica amb l'objectiu d'estudiar el problema d'embarcament de passatgers en aeronaus i conèixer quines han estat les principals contribucions i conclusions obtingudes en altres estudis, fet que permet identificar possibles àrees de millora i definir una solució que resulta en una aportació original al problema. A continuació, es defineixen dues estratègies d'embarcament especialment dissenyades per a donar resposta a aquest problema atenent a les característiques individuals dels passatgers i les relacions existents entre ells.

Un cop definides les diferents estratègies d'embarcament, s'utilitza la programació en JAVA per a implementar un programa que permet obtenir una solució al problema en funció de l'estratègia d'embarcament utilitzada. A més, es desenvolupa un simulador amb l'objectiu d'obtenir estimacions, el més realistes possible, sobre el temps total d'embarcament depenent de l'estratègia utilitzada i les condicions inicials del problema.

A partir dels valors obtinguts mitjançant la simulació, es duen a terme un conjunt de probes estadístiques que permeten estudiar el comportament i eficiència de cada una de les estratègies d'embarcament depenent de diferents condicions inicials del problema. D'altra banda, això permet comparar una de les estratègies d'embarcament proposades en aquest estudi amb altres estratègies ja existents per tal d'extreure conclusions sobre la seva eficiència en relació a la resta.

Finalment, s'exposen les principals conclusions extretes en l'estudi i s'esmenten possibles línies de treball futur per tal de facilitar una possible continuïtat en la investigació i optimització del procés d'embarcament de passatgers en aeronaus.

Secció 1: Introducció

1. Introducció i motivació

En l'actualitat cal entendre el sector del transport aeri com una indústria situada en un entorn altament competitiu en què les companyies aèries es veuen forçades a maximitzar l'efectivitat i eficiència de totes aquelles operacions sobre les quals tenen control amb l'objectiu de minimitzar costos i, d'aquesta manera, augmentar la seva rendibilitat i assegurar la seva permanència en el mercat. Tal i com mostren les últimes estimacions realitzades per organismes com ICAO[1] [2] i IATA, així com els principals manufacturers d'aeronaus Airbus [3] i Boeing [4]; la indústria del transport aeri es caracteritza per a créixer fortament amb el pas del temps i s'espera un creixement anual del volum de passatgers al voltant del 5% durant els propers 20 anys. No obstant, "l'augment de l'oferta en el transport aeri és molt més lent que el de la demanda" (Tang 2012), pel que resulta imprescindible augmentar l'eficiència de les operacions existents amb l'objectiu de minimitzar els possibles conflictes que aquest fet pugui ocasionar.

D'altra banda, resulta força evident que una aeronau només genera ingressos mentre està volant, pel que és especialment crític per a les aerolínies reduir el temps en què els avions es troben a terra, cosa que permetrà augmentar la utilització de les aeronaus permetent poder augmentar el nombre d'operacions diàries i guanyar flexibilitat per a fer front a possibles retards o imprevistos. Això implica reduir el temps d'escala als aeroports, és a dir, el temps necessari per a descarregar una aeronau un cop aquesta ha aterrat i preparar-la per a la següent operació. El fet d'augmentar l'eficiència de l'operació d'escala no és una tasca simple a causa del nombre d'actors que han de cooperar i actuar de forma conjunta per tal d'aconseguir que el procés es desenvolupi de forma fluida i eficaç. No obstant, hi ha diversos processos sobre els quals les companyies aèries poden incidir de forma directa i que poden ajudar a reduir significativament el temps total d'escala. Entre aquests, es troba l'objecte principal d'aquest estudi: el procés d'embarcament de passatgers a l'aeronau.

És possible definir el procés d'embarcament com aquell en què els passatgers d'un vol determinat aborden l'aeronau i ocupen els seients als quals han estat assignats després d'haver dipositat l'equipatge de mà, en cas de portar-ne, en els compartiments adreçats a tals efectes. Altrament, s'entén per temps total d'embarcament aquell comprès entre el primer i últim passatger en ocupar el seu seient dins de la cabina de passatgers de l'aeronau. El procés d'embarcament resulta especialment crític dintre del conjunt de processos compresos en l'operació d'escala ja que, a part de ser un dels processos que determinen el final de l'escala, sol requerir d'un temps força elevat que implica la prolongació del temps total de l'operació. Van Landeghem i Beuselinck (2000) determinaren que el temps mig de l'operació d'escala es trobava comprès entre els 30 i 60 minuts, i que gran part d'aquest temps podia atribuir-se al procés d'embarcament de passatgers.

En el present projecte es pretén tractar el problema d'embarcament de passatgers en aeronaus consistent en definir l'ordre en què els passatgers aborden una aeronau de manera que el temps total d'embarcament sigui mínim i, per tant, resulti en una reducció del temps total d'escala. Nyquist i McFadden (2008) estimaren que el fet de reduir 1 minut el temps d'escala d'una aeronau suposava, per a una companyia aèria, un estalvi aproximat de 22€. Aquest fet implica que, per exemple, el fet de reduir en 1 minut el temps d'escala de tots els vols d'una aerolínia que compta amb 300 vols diaris implica un estalvi anual de 2.409.000€.

Finalment, una millora qualitativa del procés d'embarcament de passatgers resulta en un benefici mutu entre la companyia aèria i els propis usuaris de la companyia, els quals percebran una millora en la qualitat del servei i tracte a bord.

2. Objectius

2.1 Objectius globals

Entenent una estratègia d'embarcament com una metodologia organitzada i seqüencial d'embarcar els passatgers en una aeronau, l'objectiu principal del projecte serà dissenyar una metodologia que permeti definir una estratègia d'embarcament òptima que resulti en una minimització del temps total d'embarcament en funció de les característiques dels passatgers d'un vol determinat així com les relacions existents entre aquests.

D'aquesta manera, en primer lloc es realitzarà una revisió de la literatura científica existent per tal d'estudiar el problema d'embarcament de passatgers en aeronaus i conèixer les principals aportacions sobre la matèria, cosa que permetrà identificar possibles oportunitats i àrees de millora. A més, també es pretén estudiar i analitzar si, depenent de les característiques dels passatgers d'un vol determinat, algunes estratègies resulten ser més efectives que d'altres, pel que convindria que les companyies aèries empressin diferents metodologies d'embarcament depenent dels tipus de passatgers en cada un dels seus vols. També s'estudiarà quins són els factors que perjudiquen el rendiment de cada una de les estratègies d'embarcament per tal de determinar en quina mesura afecten les característiques dels passatgers en l'èxit d'aquestes.

En gran part de la literatura científica existent s'analitza el problema d'embarcament de passatgers mitjançant models que pretenen optimitzar i determinar quina és la millor estratègia d'embarcament a ser adoptada per les companyies aèries. No obstant, "en cap d'aquests estudis es té en compte les propietats variables del moviment dels passatgers ni les seves característiques individuals de forma

explícita"(Tang et al, 2012). A més, en el present projecte es pretén determinar quina és l'assignació òptima de passatgers a seients tenint en compte la localització dels seients dintre de l'aeronau i les característiques individuals dels passatgers en cada vol així com les relacions existents entre aquests.

També és important destacar que les solucions proposades intentaran tindre en compte la seva capacitat per a ser aplicades a l'operativa real de manera que sigui possible reduir el temps total d'embarcament sense que això suposi afectar greument la satisfacció dels passatgers, la qualitat del servei i eviti qualsevol tipus de discriminació. A més, "les estratègies d'embarcament haurien de ser robustes sota els efectes de diferents tipus de pertorbacions com poden ser els retards dels passatgers, les dimensions de l'aeronau i els nivells d'ocupació" (Audenaert et al, 2009). Tal i com varen concloure P. Ferrari i K. Negel (2005) en el seu estudi sobre la solidesa de les estratègies d'embarcament eficients, "les estratègies eficients tendeixen a mantenir les persones properes separades unes de les altres", pel que posen poc èmfasi en el impacte que tindria la seva implementació a la vida real. Aquest últim fet demostra que les característiques dels passatgers són un factor imprescindible que cal tenir en compte per a garantir l'èxit d'una estratègia d'embarcament determinada.

Finalment, cal remarcar que el problema d'embarcament de passatgers en aeronaus és d'especial interès per a totes aquelles companyies aèries que operen vols de curt i mig recorregut i que, a més, depenen especialment de l'eficiència de les seves operacions. Això implica que la optimització del procés d'embarcament resulti d'especial interès per a les companyies "Low-Cost" i és per aquesta raó que l'estudi es centrarà en l'avaluació de diferents estratègies d'embarcament en aeronaus d'un únic passadís com poden ser l'Airbus a320 i el Boeing 737.

2.2 Objectius parcials

Per tal de poder assolir els objectius globals del projecte caldrà definir una sèrie d'objectius parcials que permetin poder arribar a determinar quina és l'estratègia d'embarcament més eficient segons les característiques individuals dels passatgers en cada vol. A continuació s'especifiquen els objectius parcials del projecte:

1. Estudiar i identificar les estratègies d'embarcament més utilitzades en la indústria.
2. Seleccionar i definir aquelles estratègies d'embarcament que es considerin més interessants per a estudiar i avaluar en el projecte així com definir els diferents

atributs dels passatgers que es tindrà en compte a l'hora d'implementar l'estratègia de resolució del problema i el model de simulació.

3. Definir i dissenyar de forma conceptual l'estratègia de resolució del problema, obtenir les dades i inputs necessaris i, posteriorment, implementar-ne l'algoritme mitjançant el llenguatge de programació JAVA.

4. Un cop construït l'algoritme de resolució del problema, avaluar cada una de les estratègies d'embarcament sota diferents instàncies del problema (escenaris).

5. Realitzar un estudi comparatiu sobre el rendiment i eficiència de les diverses estratègies d'embarcament avaluades per a cada un dels escenaris definits anteriorment i els resultats obtinguts en altres estudis.

Els objectius definits anteriorment permetran conèixer, en primer lloc, quines són les pràctiques més utilitzades per les companyies aèries per tal d'identificar possibles oportunitats o àrees de millora. A continuació es desenvoluparà un programa que permetrà obtenir solucions al problema d'embarcament de passatgers en aeronaus en funció de diverses estratègies diferents.

3. Metodologia

Per tal de poder assolir els objectius exposats anteriorment, resulta necessari definir una metodologia de treball que serveixi com a estructura a seguir al llarg del desenvolupament del projecte. A continuació s'especifiquen els diferents passos que se seguirà per tal de dur a terme el projecte de forma satisfactòria:

1- Revisió de la literatura científica existent sobre el problema amb l'objectiu de conèixer els diferents estudis realitzats, metodologies emprades i conclusions. La revisió de la literatura existent permetrà conèixer l'estat de l'art sobre la matèria i identificar possibles millores per tal de trobar una solució que resulti en una aportació original als treballs ja realitzats.

2- Recerca sobre les estratègies d'embarcament més utilitzades per les companyies aèries. Consistirà en realitzar un estudi comparatiu sobre les estratègies d'embarcament utilitzades per diverses companyies aèries i proposades en estudis anteriors. D'altra banda, es pretendrà conèixer quines són les estratègies d'embarcament més usuals en la indústria.

3- Definició i implementació de l'estratègia de resolució del problema en el llenguatge de programació JAVA així com dels diferents escenaris que es tindrà en compte en el projecte.

4- Anàlisi de les dades obtingudes a través de la simulació i realització de tests ANOVA.

5- Extracció de conclusions en funció dels resultats obtinguts en cada escenari i per a cada estratègia d'embarcament avaluada en l'estudi.

4. Planificació temporal

A continuació, en la figura 4.1 es mostra una taula amb una planificació orientativa de les diferents tasques a desenvolupar al llarg del projecte així com les diverses fites i dates d'entrega estimades:

Tasques	Inici	Fi
Entrega del pla de treball	**08-feb-13**	**13-feb-13**
Revisió de l'estat de l'art	**14-feb-13**	**06-mar-13**
Recopilació d'informació i idees	14-feb-13	16-feb-13
Introducció del projecte	17-feb-13	22-feb-13
Definició del problema a tractar	23-feb-13	01-mar-13
Definició d'objectius del problema	02-mar-13	06-mar-13
Recerca i definició d'estratègies d'embarcament	**07-mar-13**	**24-mar-13**
Definició de les diferents estratègies existents	07-mar-13	10-mar-13
Comparativa sobre diferents estratègies	16-mar-13	20-mar-13
Aportació original sobre noves estratègies	21-mar-13	24-mar-13
Anàlisi sobre factors causants de retrassos en l'embarcament de PAX	**27-mar-13**	**06-abr-13**
Recerca	27-mar-13	29-mar-13
Explicació sobre els diferents factors	30-mar-13	06-abr-13
Entrega de la primera versió de la memòria	**06-abr-13**	**06-abr-13**
Modelatge i simulació	**07-abr-13**	**02-jun-13**
Definició del model conceptual i escenaris	07-abr-13	14-abr-13
Modelat d'inputs del model	17-abr-13	19-abr-13
Implementació del model i escenaris	20-abr-13	11-may-13
Execució de les simulacions	12-may-13	12-may-13
Recopilació de les dades obtingudes	15-may-13	16-may-13
Anàlisi de "performance" de cada escenari	17-may-13	19-may-13
ANOVA tests	22-may-13	23-may-13
Anàlisi de dades	24-may-13	26-may-13
Conclusions de cada escenari	29-may-13	02-jun-13
Entrega de la memòria final	**13-jun-13**	**20-jun-13**

Fig. 4.1 Taula de planificació temporal del projecte

Secció 2: Conceptes bàsics i estat actual del tema

5. Estratègies d'embarcament

5.1 Introducció

Aquest apartat té per objectiu descriure i definir les principals estratègies d'embarcament proposades en estudis anteriors i utilitzades per les companyies aèries per tal de donar a conèixer i exemplificar gràficament en què consisteixen i quines són les seves fortaleses i debilitats. D'altra banda, el fet de conèixer les fortaleses i debilitats de cada estratègia d'embarcament permetrà definir una nova estratègia que sigui el més realista i eficient possible i que es beneficiï de les fortaleses de les estratègies ja existents.

Tal i com s'ha definit anteriorment, una estratègia d'embarcament consisteix en una metodologia que permet definir, de manera seqüencial, l'ordre en què els passatgers accedeixen a l'aeronau per tal d'ocupar els seients als quals han estat assignats. Normalment, la creació d'una estratègia d'embarcament implica l'assignació de passatgers a grups d'embarcament per tal de que aquests ocupin determinades zones de la cabina de l'aeronau en un ordre establert per la companyia aèria amb l'objectiu de minimitzar el temps total d'embarcament. A més, la majoria d'estratègies d'embarcament es basen en la localització dels seients als quals han estat assignats els passatgers, pel que s'assumeix que, abans de realitzar l'assignació de passatgers a grups d'embarcament, s'ha assignat un seient a cada passatger. Per molt evident que pugui semblar, és important remarcar aquest últim fet ja que no necessàriament totes les estratègies d'embarcament assumeixen que s'ha dut a terme una assignació prèvia de seients a passatgers. D'altra banda, hi ha estratègies, com és el cas de l'estratègia que es proposarà en aquest estudi, que incorporen l'assignació de seients a passatgers com una fase de la pròpia estratègia d'embarcament.

Finalment, és important destacar que no totes les estratègies d'embarcament es basen en els mateixos factors, el qual implica que estratègies que poden resultar molt adequades per a certes companyies aèries no ho siguin per d'altres. Una bona mostra d'aquest fet són, per exemple, aquelles estratègies d'embarcament en què s'assigna passatgers a grups d'embarcament en funció del nivell de fidelitat o "estatus" en relació a la companyia. Si bé aquest tipus d'estratègies d'embarcament poden resultar molt adequades per a companyies aèries tradicionals o "full service" en què s'ofereix una gamma de productes molt variada, és molt possible que tinguin poca cabuda en companyies que ofereixen un únic producte, com és el cas de la majoria de companyies "Low Cost".

5.2 Definició de les principals estratègies d'embarcament

Estratègia aleatòria o *Random*:

Tal i com el propi nom indica, l'estratègia d'embarcament aleatòria (veure figura 5.1) consisteix en no establir cap ordre durant tot el procés d'embarcament. Això implica que els passatgers no són assignats a grups d'embarcament, sinó que accedeixen a la cabina de l'aeronau per a ocupar els seients als quals han estat assignats en el mateix ordre en què s'han disposat a la cua de les portes d'embarcament.

Si bé pot semblar que es tracta d'una estratègia força poc eficient pel fet que no hi ha cap tipus de control en l'ordre en què els passatgers accedeixen a l'aeronau, fet que implica que sigui impossible controlar el nombre total d'interferències o conflictes entre els passatgers, estudis anteriors demostren que es tracta d'una estratègia molt eficient en termes del temps total d'embarcament. Possiblement, una de les raons que expliquen l'eficiència de l'estratègia aleatòria és el fet que permet maximitzar el nivell d'activitat dintre de l'aeronau ja que, al haver-hi un únic grup d'embarcament, és molt possible que diversos passatgers estiguin ocupant els seients als quals han estat assignats de forma simultània. A més, es tracta d'una estratègia molt robusta pel fet que la seva eficiència no depèn del nivell de puntualitat dels passatgers a les portes d'embarcament donat que aquests embarquen l'aeronau sense haver d'esperar a ser cridats per la companyia aèria. Aquest no és el cas d'altres estratègies en què l'ordre d'embarcament ve marcat per la creació de diversos grups d'embarcament. En aquests casos s'assumeix que tots els passatgers seran a la porta d'embarcament en el moment en què el grup al qual han estat assignats sigui cridat per a accedir a la cabina de l'aeronau.

A	B	C		D	E	F
1	1	1	PASSADÍS	1	1	1
1	1	1		1	1	1
1	1	1		1	1	1
1	1	1		1	1	1
1	1	1		1	1	1
1	1	1		1	1	1
1	1	1		1	1	1
1	1	1		1	1	1
1	1	1		1	1	1
1	1	1		1	1	1
1	1	1		1	1	1
1	1	1		1	1	1

Fig. 5.1 Seqüència d'embarcament establerta en l'estratègia aleatòria o R*andom*

L'estratègia d'embarcament aleatòria és especialment popular entre les companyies aèries "Low Cost" degut a la seva gran eficiència i al fet que es tracta d'una estratègia molt adequada per a companyies que operen amb aeronaus d'una única configuració de cabina ("full economy"). A tall d'exemple, algunes de les aerolínies que utilitzen aquesta estratègia d'embarcament són Jet2, JetBlue, Ryanair i WOW air.

En darrer terme, convé subratllar que no s'ha de confondre l'estratègia d'embarcament aleatòria amb l'estratègia *free-for-all* o *free-seating*. Aquesta última es diferencia de l'estratègia aleatòria pel fet que els passatgers no són assignats a cap seient, sinó que accedeixen a la cabina de l'aeronau en l'ordre en què s'han disposat a la cua de les portes d'embarcament i ocupen el seient que més els plau. Un bon exemple és el cas de Southwest airlines, en què els passatgers són assignats a tres possibles grups (A, B o C) en funció de l'antelació amb la qual han realitzat la reserva del seu vol i ocupen el seient que troben més convenient.

Estratègia *Back-to-front*:

En l'estratègia d'embarcament *Back-to-front* els passatgers són cridats per embarcar l'aeronau en ordre de fila descendent. D'aquesta manera s'aconsegueix embarcar l'aeronau començant per les últimes files amb l'objectiu d'eliminar qualsevol conflicte de passadís, fet que permet minimitzar el temps total d'embarcament. No obstant, per tal d'eliminar els conflictes de passadís en la seva totalitat caldria ordenar els passatgers a la cua d'embarcament en funció de la fila a la qual han estat assignats, fet que converteix l'estratègia *Back-to-front* per files en una estratègia eminentment teòrica donada la seva poca capacitat de ser implementada en l'operativa real.

Degut que no és possible ordenar els passatgers a la cua de les portes d'embarcament en funció de la fila a la qual han estat assignats, l'estratègia d'embarcament *Back-to-front* per blocs permet embarcar determinades zones de l'aeronau mitjançant l'assignació de grups d'embarcament a passatgers en funció de la fila del seient al qual han estat assignats. Això permet definir l'ordre en què s'embarcarà cada una de les zones de l'aeronau, tot i que no serà possible garantir l'eliminació total dels conflictes de passadís pel fet que l'embarcament dintre de cada grup serà aleatori.

En la següent figura (5.2) es mostra un exemple de l'ordre d'embarcament establert en l'estratègia *Back-to-front* per files (esquerra) i per blocs (dreta):

A	B	C		D	E	F
12	12	12		12	12	12
11	11	11		11	11	11
10	10	10		10	10	10
9	9	9		9	9	9
8	8	8		8	8	8
7	7	7		7	7	7
6	6	6		6	6	6
5	5	5		5	5	5
4	4	4		4	4	4
3	3	3		3	3	3
2	2	2		2	2	2
1	1	1		1	1	1

PASSADÍS

A	B	C		D	E	F
4	4	4		4	4	4
4	4	4		4	4	4
4	4	4		4	4	4
3	3	3		3	3	3
3	3	3		3	3	3
3	3	3		3	3	3
2	2	2		2	2	2
2	2	2		2	2	2
2	2	2		2	2	2
1	1	1		1	1	1
1	1	1		1	1	1
1	1	1		1	1	1

PASSADÍS

Fig. 5.2 Seqüència d'embarcament establerta en l'estratègia *Back-to-front* per files (esquerra) i per blocs (dreta)

Com a exemple, algunes de les companyies aèries que utilitzen l'estratègia *Back-to-front* per blocs en la seva operativa són: Air Canada, British Airways, Virgin Atlantic i Vueling.

Estratègia *Front-to-back*:

De manera inversa a l'estratègia *Back-to-front*, l'estratègia *Front-to-back* (veure fig. 5.3) consisteix en embarcar l'aeronau per ordre de fila ascendent. En aquest cas, pels mateixos motius exposats anteriorment, els passatgers són dividits en diversos grups d'embarcament en funció de la fila corresponent al seient al qual han estat assignats i procedeixen a embarcar l'aeronau en l'ordre establert per a la companyia aèria. En la següent figura s'exemplifica gràficament la seqüència d'embarcament marcada per l'estratègia *Front-to-back* per files i per blocs.

A	B	C		D	E	F
1	1	1		1	1	1
2	2	2		2	2	2
3	3	3		3	3	3
4	4	4		4	4	4
5	5	5		5	5	5
6	6	6		6	6	6
7	7	7		7	7	7
8	8	8		8	8	8
9	9	9		9	9	9
10	10	10		10	10	10
11	11	11		11	11	11
12	12	12		12	12	12

PASSADÍS

A	B	C		D	E	F
1	1	1		1	1	1
1	1	1		1	1	1
1	1	1		1	1	1
2	2	2		2	2	2
2	2	2		2	2	2
2	2	2		2	2	2
3	3	3		3	3	3
3	3	3		3	3	3
3	3	3		3	3	3
4	4	4		4	4	4
4	4	4		4	4	4
4	4	4		4	4	4

PASSADÍS

Fig. 5.3 Seqüència d'embarcament establerta en l'estratègia *Front-to-back* per files (esquerra) i per blocs (dreta)

Convé destacar que l'estratègia *Front-to-back* sol utilitzar-se amb conjunció amb l'estratègia *Back-to-front* pel fet que resulta especialment útil per a totes aquelles companyies que ofereixen diverses classes en els seus vols. Això permet establir prioritats en l'ordre en què els passatgers embarquen l'aeronau en funció de la classe en què viatgen o el nivell d'estatus de fidelitat que mantenen amb la companyia aèria. En definitiva, la combinació de les dues estratègies d'embarcament es podria resumir en una única estratègia consistent en embarcar l'aeronau per blocs.

A causa dels motius exposats anteriorment, l'embarcament per blocs resulta molt útil per a totes aquelles companyies aèries que divideixen la cabina de l'aeronau en diferents zones depenent de la classe en què viatgen els passatgers. Un bon exemple d'això és el cas d'American Airlines (fig. 5.4), en què els primers passatgers en embarcar són els que tenen estatus d'"Elite", "first" o "business", tal i com es mostra en la figura següent:

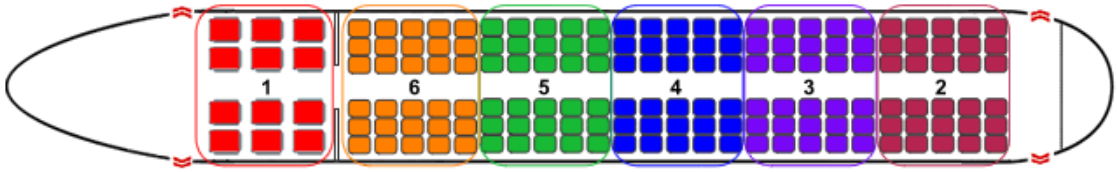


Fig. 5.4 Seqüència d'embarcament utilitzada per American Airlines

Estratègia *WILMA*:

L'estratègia d'embarcament *WILMA* ("windows, middle, aisle") o *Outside-in* té com a objectiu principal eliminar els possibles conflictes de seient causats entre passatgers assignats a la mateixa fila de l'aeronau. Per tal d'aconseguir-ho, els passatgers són assignats a grups d'embarcament en funció del seient al qual han estat assignats i embarquen l'aeronau en el següent ordre:

1- Passatgers amb seients situats a les finestres (seients A o F).
2- Passatgers amb seients situats al centre de cada mitja fila (seients B o E).
3- Passatgers amb seients situats al costat del passadís (seients C o D).

Si bé és cert que l'eliminació dels conflictes de seient pot resultar en una reducció significativa del temps total d'embarcament, l'estratègia *WILMA* resulta difícilment aplicable a l'operativa real pel fet que no té en compte les relacions existents entre els passatgers. Això es deu a que l'ordre d'embarcament i el criteri d'assignació de passatgers a grups d'embarcament es basa, únicament, en el seient de cada passatger. D'aquesta manera, passatgers que viatgin junts hi hagin estat assignats a seients

contigus es veuran forçats a separar-se durant l'embarcament, fet que resultarà en un nivell d'insatisfacció molt elevat per part dels passatgers.

La figura següent (5.5) il·lustra l'ordre en que els passatgers embarquen l'aeronau d'acord amb l'estratègia *WILMA*:

A	B	C		D	E	F
1	2	3		3	2	1
1	2	3		3	2	1
1	2	3		3	2	1
1	2	3		3	2	1
1	2	3		3	2	1
1	2	3		3	2	1
1	2	3		3	2	1
1	2	3		3	2	1
1	2	3		3	2	1
1	2	3		3	2	1
1	2	3		3	2	1
1	2	3		3	2	1

PASSADÍS

Fig. 5.5 Seqüència d'embarcament establerta en l'estratègia *WILMA*

Estratègia *Reverse Pyramid*:

L'estratègia d'embarcament *Reverse Pyramid* va ser dissenyada per Menkes Van den Briel en un estudi realitzat l'any 2005. Van den Briel s'adonà que les estratègies d'embarcament per blocs solien deixar zones de la cabina de l'aeronau infrautilitzades, fet que resultava en un nivell d'ociositat força elevat. Per tal de resoldre aquest problema, l'estratègia *Reverse Pyramid* combina aspectes d'estratègies ja existents amb l'objectiu d'embarcar l'aeronau des dels seients exteriors cap als interiors sense desaprofitar zones de la cabina de l'aeronau. Això s'aconsegueix assignant, en un mateix grup, passatgers amb seients situats en diferents zones de l'aeronau, fet que permet maximitzar el nivell d'activitat en totes les zones de l'avió. A més, al embarcar l'aeronau des dels seients exteriors cap als interiors s'aconsegueix neutralitzar possibles conflictes de seient entre els passatgers, fet que permet minimitzar el temps total d'embarcament. D'alguna manera, es podria considerar l'estratègia *Reverse Pyramid* com una combinació de l'estratègia *Back-to-front* per blocs i l'estratègia *WILMA* pel fet que combina l'embarcament en ordre de fila descendent i des dels seients exteriors cap als interiors.

Mentre que diversos estudis coincideixen en concloure que l'estratègia d'embarcament *Reverse Pyramid* és altament eficient i permet obtenir temps totals d'embarcament molt baixos, un dels principals inconvenients és la seva baixa

aplicabilitat a l'operativa real. Es tracta d'una estratègia d'embarcament poc aplicable pel fet que, de la mateixa manera que l'estratègia *WILMA*, no té en compte les relacions existents entre els passatgers. El fet d'embarcar l'aeronau des dels seients exteriors cap als interiors ocasionarà que passatgers que viatgin junts i hagin estat assignats a seients contigus hagin de separar-se durant l'embarcament.

En la figura següent (5.6) es mostra un exemple de l'ordre d'embarcament establert per l'estratègia:

A	B	C		D	E	F
3	4	5		5	4	3
3	4	5		5	4	3
3	4	5		5	4	3
2	3	5		5	3	2
2	3	5		5	3	2
2	3	5		5	3	2
1	3	5		5	3	1
1	2	4		4	2	1
1	2	4		4	2	1
1	2	4		4	2	1
1	2	4		4	2	1
1	2	4		4	2	1

PASSADÍS

Fig. 5.6 Seqüència d'embarcament establerta en l'estratègia *Reverse Pyramid*

Estratègia de Steffen i *Kautzka 3*:

A partir de les conclusions extretes en estudis anteriors en què s'observà que, a part de les interferències causades entre els passatgers, un dels factors que més afecten al temps total d'embarcament és el temps necessari per a dipositar l'equipatge de mà, Steffen va desenvolupar l'any 2008 una nova estratègia d'embarcament amb l'objectiu de maximitzar el nombre de passatgers que carreguen l'equipatge de forma simultània. Per tal d'aconseguir-ho, Steffen va considerar que la forma òptima de maximitzar el nivell d'activitat simultània a la cabina de l'aeronau era mitjançant una combinació de l'estratègia *WILMA* i *Back-to-front*, deixant una fila de separació entre els passatgers de cada grup d'embarcament per a garantir que hi hagués espai suficient per a dur a terme l'embarcament de forma còmode. D'aquesta manera, l'assignació de passatgers a grups d'embarcament es realitza en funció del seient i la fila assignada a cada passatger, tal i com es mostra a continuació (fig. 5.7):

A	B	C		D	E	F
3	7	11		12	8	4
1	5	9		10	6	2
3	7	11		12	8	4
1	5	9		10	6	2
3	7	11	PASSADÍS	12	8	4
1	5	9		10	6	2
3	7	11		12	8	4
1	5	9		10	6	2
3	7	11		12	8	4
1	5	9		10	6	2
3	7	11		12	8	4
1	5	9		10	6	2

Fig. 5.7 Seqüència d'embarcament establerta en l'estratègia proposada per Steffen

Adonant-se de que l'estratègia proposada per Steffen no permetia mantenir junts a passatgers que viatgen en parelles, Cimler va desenvolupar, l'any 2012, una adaptació de l'estratègia que anomenà *Kautzka 3* (veure fig. 5.8). En aquest cas, cada grup d'embarcament està format per parelles de passatgers assignats a seients contigus en la mateixa fila, tal i com es mostra a continuació:

A	B	C		D	E	F
3	3	6		6	4	4
1	1	5		5	2	2
3	3	6		6	4	4
1	1	5		5	2	2
3	3	6		6	4	4
1	1	5	PASSADÍS	5	2	2
3	3	6		6	4	4
1	1	5		5	2	2
3	3	6		6	4	4
1	1	5		5	2	2
3	3	6		6	4	4
1	1	5		5	2	2

Fig. 5.8 Seqüència d'embarcament establerta en l'estratègia *Kautzka 3*

5.3 Interferències i conflictes

Durant tot el procés d'embarcament, donat que els passatgers accedeixen a l'aeronau de forma conjunta i a causa de les limitacions d'espai establertes per les dimensions de la cabina, és possible que es produeixin un conjunt d'interferències o conflictes entre

els passatgers. Aquestes interferències es consideren importants pel fet que solen ser la causa de retards en el temps total d'embarcament, i tenen com a conseqüència un allargament innecessari del procés d'embarcament i una percepció negativa per part dels passatgers del servei ofert per la companyia aèria. Concretament, és possible distingir entre dos tipus de conflictes entre els passatgers: conflictes de passadís i conflictes de seient.

Els conflictes de passadís tenen lloc quan l'accés d'un passatger al seient al qual ha estat assignat es troba obstaculitzat per un altre passatger situat al passadís de la cabina de l'aeronau. En la figura 5.9 es mostra un exemple en què el pas del passatger 2, que ha estat assignat al seient 6A, es troba obstaculitzat pel passatger 1, el qual ha estat assignat al seient 4F i es troba desant l'equipatge de mà. En aquest cas, el passatger 2 es veu obligat a esperar a que el passatger 1 ocupi el seu seient per tal de poder avançar i arribar a la fila del seient al qual ha estat assignat, fet que té com a conseqüència un retard en el temps d'embarcament del passatger 2.

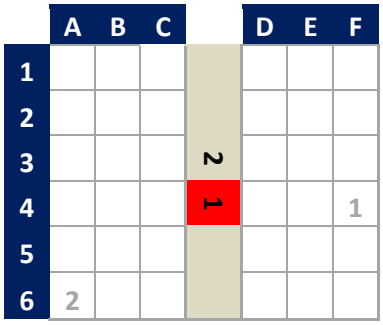


Fig. 5.9 Exemple d'un conflicte de passadís

Els conflictes de seient es donen en aquells casos en que els passatgers no poden accedir al seient al qual han estat assignats a causa d'altres passatgers que es troben asseguts a la mateixa fila, tal i com es mostra en el següent exemple (fig. 5.10):

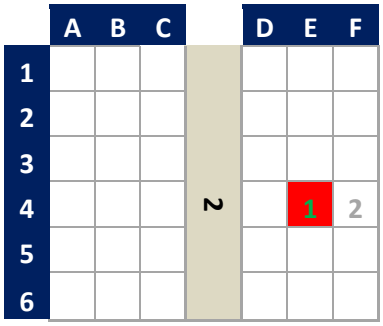


Fig. 5.10 Exemple d'un conflicte de passadís

En aquest cas, el passatger 1 haurà d'aixecar-se per tal de permetre que el passatger 2 ocupi el seient al qual ha estat assignat.

5.4 Estratègies d'embarcament eficients

A partir de la definició i anàlisi de cada una de les estratègies d'embarcament esmentades anteriorment i les conclusions extretes en altres estudis és possible identificar quins són els factors que cal tenir en compte per a dissenyar una estratègia d'embarcament eficient que permeti minimitzar el temps total d'embarcament. Concretament, els principals punts que cal tenir en compte per a obtenir estratègies eficients són:

- Minimitzar el nombre total de conflictes de passadís originats per passatgers que impedeixen que altres passatgers accedeixin al seient al qual han estat assignats.

- Minimitzar el nombre total de conflictes de seient.

- Maximitzar el nivell d'activitat dintre de la cabina de l'aeronau amb l'objectiu de permetre que el màxim nombre de passatgers realitzin tasques de forma simultània, fet que permetrà aprofitar al màxim l'espai disponible a la cabina i reduir el nivell d'ociositat dels passatgers.

No obstant, a l'hora de dissenyar una estratègia d'embarcament eficient és important tenir en compte la seva capacitat per a ser aplicada a l'operativa real. Això implica prendre consciència sobre els lligams i relacions existents entre els passatgers, així com la percepció que aquests tindran en relació al servei ofert per la companyia aèria i tracte rebut per part d'aquesta. Les principals restriccions que cal respectar per tal d'assegurar l'aplicabilitat de qualsevol estratègia d'embarcament són les següents:

- No és possible ordenar individualment els passatgers a la cua d'embarcament ni definir, de forma individual, l'ordre en què aquests accedeixen a la cabina de l'aeronau. Això implica que, dintre de cada grup d'embarcament, l'ordre en què els passatgers accedeixen a l'aeronau serà desconegut i incontrolable, fet que impossibilita l'eliminació total dels conflictes de passadís.

- No és possible separar a passatgers que viatgen conjuntament en cap moment del procés d'embarcament.

- No és possible realitzar discriminacions de ningun tipus entre els passatgers.

6. Revisió de la literatura científica

El problema d'embarcament de passatgers en aeronaus ha estat objecte d'un nombre molt elevat d'estudis al llarg de les últimes dècades. Les següents pàgines constitueixen una revisió de la literatura científica existent, on s'exposen, de forma resumida, els principals estudis i aportacions sobre la matèria. Això permetrà identificar possibles àrees de millora i, d'aquesta manera, realitzar un estudi que resulti en una aportació original al problema.

L'any 1998 Marelli, Mattocks i Merry, junt amb l'empresa Boeing, desenvoluparen un model de simulació orientat a esdeveniments discrets anomenat *Boeing Passenger Enplane/Deplane Simulation (PEDS)* per tal d'avaluar diferents estratègies d'embarcament i configuracions en la cabina d'un Boeing 757-200. En l'estudi es va concloure que el temps total d'embarcament utilitzant una única porta d'accés a l'aeronau i una estratègia d'embarcament tradicional (i.e. per blocs o per files) era, aproximadament, de 26 minuts. A més, també observaren que utilitzar dues portes per a embarcar l'aeronau permetia reduir el temps d'embarcament en 5 minuts, mentre que si s'utilitzaven dues portes i una estratègia d'embarcament consistent en embarcar primer els passatgers de les finestres era possible reduir el temps total d'embarcament en 17 minuts. Conseqüentment, Marelli et al. determinaren que l'estratègia d'embarcament que minimitzava el temps total era aquella consistent en embarcar l'aeronau des dels seients exteriors cap als interiors (i.e. de les finestres cap al passadís) ja que això permetia neutralitzar gran part de les interferències entre els passatgers.

Van Landeghem i Beuselinck (2002), de la universitat de Ghent, exploraren diverses estratègies d'embarcament per tal de determinar, mitjançant un model de simulació amb "Arena", en quina mesura era possible reduir el temps total d'embarcament en una aeronau de 132 seients i 23 files. Els autors van obtenir dades numèriques proporcionades per l'aeroport de Brussel·les i l'aerolínia belga Sabena que utilitzaren com a inputs del model de simulació. D'especial interès resulta el fet que en l'estudi es va definir el problema d'embarcament de passatgers en aeronaus així com les diverses estratègies d'embarcament de manera formal. Tanmateix, en l'estudi s'esmenta les diferents metodologies emprades per a aplicar les estratègies d'embarcament ("Call-off systems") així com els possibles conflictes i restriccions que cal tenir en compte a l'hora de construir el model. D'alguna manera es podria considerar que l'estudi realitzat per Van Landeghem i Beuselinck va ésser pioner en el plantejament formal del problema d'embarcament de passatgers en aeronaus.

D'altra banda, de la mateixa manera que Marelli et al. , Van Landeghem i Beuselinck determinaren que l'estratègia d'embarcament més eficient consistia en embarcar els passatgers des dels seients exteriors cap als interiors (veure fig. 6.1), el qual resultava

en un temps total aproximat de 10,5 minuts. A més, observaren que el principal causant de l'aparició de congestions a l'aeronau és l'operació consistent en dipositar l'equipatge de mà i que existia un elevat grau de correlació entre el temps total d'embarcament i el temps mig d'embarcament de cada passatger. Aquest últim fet permet concloure que les estratègies d'embarcament més eficients seran aquelles que permetin l'embarcament simultani de passatgers, repartint un nivell d'activitat homogeni en tota la cabina de l'aeronau.

Front						
1	2	3		3	2	1
1	2	3		3	2	1
1	2	3		3	2	1
1	2	3		3	2	1
1	2	3		3	2	1
1	2	3		3	2	1
1	2	3		3	2	1
1	2	3		3	2	1
1	2	3		3	2	1
1	2	3		3	2	1
1	2	3		3	2	1
1	2	3		3	2	1

Fig. 6.1 Estratègia d'embarcament *WILMA*

A diferència dels estudis realitzats anteriorment, Van den Briel et al. (2003,2005) junt amb la col·laboració d'America West Airlines desenvoluparen un model de programació entera no lineal amb l'objectiu de minimitzar el nombre d'interferències que tenen lloc a la cabina de l'aeronau durant el procés d'embarcament. Es tracta, doncs, d'un nou enfocament al problema d'embarcament de passatgers en aeronaus en què s'assumeix que les interferències causades a la cabina de l'aeronau són la principal causa de la ineficiència del procés i que, per tant, reduir el nombre d'interferències tindrà com a conseqüència una reducció del temps total d'embarcament. A més, una aportació destacable és que Van den Briel dissenyà una nova estratègia d'embarcament que batejà amb el nom de *reverse pyramid* i que combina aspectes d'estratègies tradicionals (*back-to-front* i *WILMA*) amb l'objectiu de maximitzar l'espai útil a la cabina de l'aeronau durant tot el procés. La figura següent (6.2) mostra el model de *reverse pyramid* esmentat anteriorment:

			Cockpit			
Window	Middle	Aisle		Aisle	Middle	Aisle
		5		5	4	
3	4	5		5	4	3
3	4	5		5	4	3
3	4	5		5	4	3
3	3	5		5	3	3
2	3	5		5	3	2
2	3	5		5	3	2
2	3	5		5	3	2
2	3	5		5	3	2
2	3	5		5	3	2
1	3	5		5	3	1
1	3	5		5	3	1
1	3	5		5	3	1
1	3	4		4	3	1
1	2	4		4	2	1
1	2	4		4	2	1
1	2	4		4	2	1
1	2	4		4	2	1
1	2	4		4	2	1
1	2	4		4	2	1
1	2	4		4	2	1
1	2	4		4	2	1
		4		4	2	

Fig. 6.2 Model d'embarcament *Reverse Pyramid*

En l'estudi es conclou que l'estratègia que minimitza el nombre d'interferències és la de *reverse pyramid* i que el nombre de grups òptim en què cal dividir els passatgers per a realitzar l'embarcament és de quatre ja que, si hi ha menys grups, s'augmenta significativament el nombre d'interferències entre membres del mateix grup, mentre que si el nombre de grups és massa elevat l'embarcament augmenta en complexitat i esdevé difícil de controlar.

Tenint en compte les conclusions extretes en treballs anteriors, l'estudi de Bachmat et al. (2005) va posar especial èmfasi en el propi disseny de la cabina de l'aeronau, observant com factors tals com la distància entre files afectaven al temps total d'embarcament. Durant el mateix any, Ferrari i Nagel (2005) van avaluar la robustesa de diverses estratègies d'embarcament sota pertorbacions com, per exemple, l'arribada amb retard de passatgers a la porta d'embarcament. Una conclusió interessant va ser comprovar que estratègies d'embarcament tradicionals com l'estratègia *back-to-front,* resultaven beneficiades pels efectes de pertorbacions. De manera paral·lela, es va observar que aquelles estratègies que resultaren ser més eficients sota condicions d'embarcament òptimes, ho continuaven sent amb l'aparició de pertorbacions.

De forma similar a l'estudi realitzat per Van den Briel et al. (2003,2005), Bazargan (2007) va desenvolupar un model de programació lineal per a estudiar i optimitzar el procés d'embarcament d'aeronaus minimitzant el nombre d'interferències que tenen

lloc al llarg de tot el procés. En el model matemàtic es tingué en compte, principalment, dos tipus d'interferències diferents:

1. Interferència de seient: Es dóna quan un passatger ha d'aixecar-se del seu seient per a facilitar que un altre passatger pugui seure.
2. Interferència de passadís: És aquella en què un passatger bloqueja el passadís per a dipositar l'equipatge de mà, impedint que la resta de passatgers avancin cap als seus respectius seients.

Els resultats obtinguts amb CPLEX mostraren que les estratègies més eficients no només són aquelles que causen un menor nombre d'interferències sinó que també són més atractives pels passatgers ja que permeten que parelles, grups i famílies embarquin alhora. Això es deu al fet que el model incorpora un paràmetre (α) que mesura la interferència entre grups, és a dir, el nombre de passatgers que no han estat capaços d'embarcar abans que arribi el següent grup de passatgers. És per aquest motiu que Bazargan no considera que l'estratègia d'embarcament de *reverse pyramid* sigui eficient quan α=0, ja que això implica que molts dels passatgers hagin de separar-se durant l'embarcament.

L'any 2007, Inman, Jones i Thompson van realitzar diversos tests ANOVA per a determinar si existien diferències significatives en els temps mitjos d'embarcament depenent de tres estratègies d'embarcament diferents (*Back-to-front*, *WILMA* i *Random*) i tres tipus d'aeronau de diferents dimensions. Per a fer-ho, en primer lloc van desenvolupar un programa informàtic que anomenaren *QuickBoard* per a simular el procés d'embarcament en cada un dels escenaris i, a continuació, estimaren el temps mig d'embarcament en cada escenari per a poder comparar-los en el test ANOVA. Finalment es conclou que, independentment de les dimensions de l'aeronau, existeixen diferències significatives en el temps mig d'embarcament utilitzant una estratègia o una altra. Aquest fet suposa una contribució important ja que implica que, a causa de la naturalesa simètrica de la cabina, és molt probable que l'eficiència de cada estratègia d'embarcament respecte la resta es mantingui encara que variïn les dimensions de l'aeronau. A més a més, també es conclou que, per a cada tipus d'aeronau, hi ha una estratègia d'embarcament que resulta més eficient que la resta.

A partir de les conclusions extretes d'estudis realitzats anteriorment, Nyquist i McFadden (2008) estudiaren el problema d'embarcament de passatgers en aeronaus de forma teòrica. L'estudi tenia per objectiu trobar una estratègia d'embarcament eficient que permetés reduir el temps total d'embarcament sense afectar greument a la satisfacció dels passatgers. Durant el mateix any, Steffen (2008) va utilitzar un

algoritme ILS[1] anomenat *Markov Chain Monte Carlo* (MCMC) per a optimitzar el procés d'embarcament. A més, també va proposar una nova seqüència d'embarcament.

Donat que els dos estudis esmentats anteriorment resulten d'especial interès per al projecte, resulta convenient explicar-los amb més detall al final d'aquest apartat.

Wang et al (2009), combinaren la programació entera mixta no lineal (MINLP) amb l'ús d'algoritmes genètics per a determinar una estratègia d'embarcament òptima. En l'estudi es formula el problema mitjançant MINLP , definint una funció objectiu no lineal i un conjunt de restriccions matemàtiques que cal tenir en compte per a poder proposar solucions factibles. A més, les estratègies d'embarcament es defineixen mitjançant l'assignació de seients a diversos grups d'embarcament. D'aquesta manera, cada seient *(i,j)*, on *i* representa el número de fila i *j* la columna, s'assigna a un determinat grup d'embarcament *k* que pertany al conjunt de grups d'embarcament *G*. Amb el plantejament descrit anteriorment, es defineix una variable de decisió binària X_{ijk} que pren el valor 1 si el seient *(i,j)* està assignat al grup d'embarcament *k* i 0 en cas contrari. També és necessari esmentar que en la funció objectiu és pretén minimitzar el nombre total d'interferències, aplicant una penalització a cada una d'aquestes mitjançant la definició de paràmetres.

Un cop definit el model mitjançant MINLP, s'utilitza un algoritme genètic com a heurística per a trobar solucions òptimes dintre de l'espai de solucions factibles. A continuació s'utilitza la simulació per tal de contrastar i validar els resultats obtinguts amb el model analític així com per a comparar la solució proposada pel model amb les estratègies *Back-to-front* i *WILMA*. Finalment, els resultats permeten concloure que l'estratègia obtinguda mitjançant el model de programació MINLP és superior a la resta d'estratègies ja que permet minimitzar el nombre d'interferències, fet que resulta en un temps total d'embarcament mínim.

A partir de les conclusions extretes d'estudis anteriors, Audenaert et al (2009) utilitzaren un model de simulació basat en agents múltiples per tal de proposar estratègies d'embarcament que fossin robustes envers a possibles pertorbacions i, a més, no perjudiquessin la satisfacció dels passatgers. És important destacar que el fet d'utilitzar aquest tipus de simulació, a diferència d'altres estudis realitzats anteriorment, permet definir trets individuals per a cada un dels passatgers, el que fa possible l'avaluació i definició d'estratègies d'embarcament basades en les característiques d'aquests. És precisament per aquest motiu que en l'estudi s'avaluaren estratègies que, per tal de minimitzar el nombre d'interferències, ordenen als passatgers en funció de la velocitat amb la que es desplacen pel passadís de l'aeronau i el nombre de peces d'equipatge de mà. Un altre punt important és que

[1] ILS són les sigles, en anglès, de *Iterated Local Search.* Es tracta d'un procés iteratiu que permet millorar una solució inicial mitjançant la realització de pertorbacions sobre aquesta.

cada estratègia d'embarcament és avaluada sota diversos factors d'ocupació. S'observa que, si el factor d'ocupació és superior al 66%, el fet d'utilitzar estratègies d'embarcament no aleatòries permet millorar de forma significativa el temps total d'embarcament.

A més a més, s'estudià els efectes de l' incompliment dels passatgers envers a cada estratègia d'embarcament[2] amb l'objectiu de determinar-ne la seva robustesa. Per fer-ho, s'assumí que quan un passatger no compleix amb la política d'embarcament, aquest embarca l'aeronau en un instant aleatori, ja sigui abans o després de l'instant en que hauria d'haver embarcat d'acord amb l'estratègia vigent. Després d'avaluar cada estratègia sota diferents nivells de compliment (100%, 66% i 33%), tal i com es mostra en la figura 6.3 ,s'observà que com menor és el nivell de compliment, el rendiment de les estratègies d'embarcament tendeix a assimilar-se als resultats obtinguts utilitzant un embarcament aleatori.

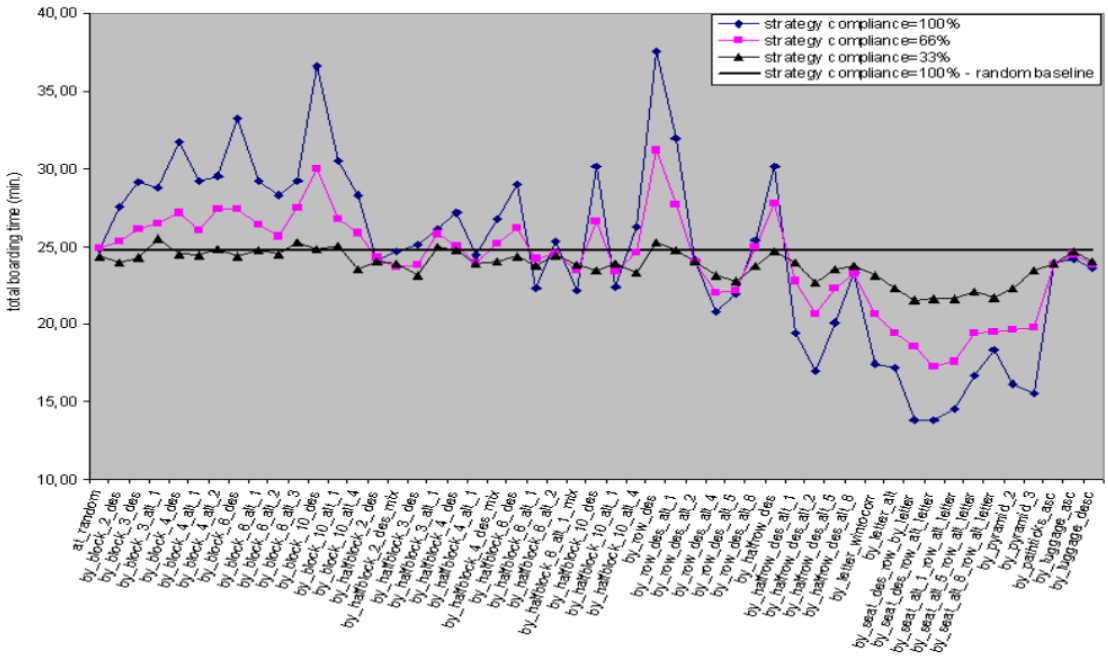


Fig. 6.3 Gràfic sobre l'efecte de diferents nivells de compliment en el temps total d'embarcament

Finalment, tenint en compte tots els factors descrits anteriorment i després d'analitzar els resultats de la simulació obtinguts per a cada escenari, es va concloure que les estratègies més eficients són les de *reverse pyramid* i aquelles que realitzen l'embarcament per seients de forma individual. No obstant, s'observà que aquestes estratègies no tenen en compte la satisfacció dels passatgers ja que tendeixen a separar grups i famílies, pel que resulta més convenient utilitzar estratègies basades en les característiques dels passatgers.

[2] La major causa d'incompliment d'estratègies d'embarcament per part dels passatgers es dóna en el moment en que aquests arriben tard a la porta d'embarcament o ocupen seients als quals no han estat assignats.

En la mateixa línea que l'estudi realitzar per Wang et al, Soolaki et al (2011) van fer ús d'algoritmes genètics (GA) per a solucionar el problema d'embarcament de passatgers en aeronaus. En l'estudi es va desenvolupar un model de programació entera lineal amb l'objectiu de trobar l'assignació òptima que permet minimitzar el nombre d'interferències i, conseqüentment, el temps total d'embarcament.

De la mateixa manera que Bazargan (2007), en el model de programació s'introdueix un paràmetre α per a mesurar el nombre de passatgers que no han estat capaços d'embarcar abans que arribi el següent grup de passatgers; el que permet tenir en compte de forma explícita les interferències causades entre passatgers de diferents grups durant tot el procés. A continuació, se soluciona el problema d'optimització mitjançant LINGO,CPLEX i GA per a diferents mides de grups d'embarcament. S'observa que l'eficiència de les solucions proposades mitjançant GA creix a mesura que creix el nombre de grups d'embarcament, de manera que en una aeronau de 23 files i utilitzant 5 grups d'embarcament els resultats obtinguts mitjançant GA són millors que els proposats per LINGO o CPLEX. També es demostra que la seqüència d'embarcament i assignació de seients obtinguda mitjançant el model de programació lineal i la utilització d'algorismes genètics resulta ser més eficient que si s'utilitzen altres estratègies d'embarcament eficients com és el cas de la *reverse pyrmaid.*

A diferència dels estudis exposats anteriorment, Steffen i Hotchkiss (2011) van realitzar una sèrie de proves amb passatgers voluntaris en un fuselatge de Boeing 757 per tal de mesurar i estimar de forma empírica el temps total d'embarcament utilitzant diferents estratègies. Es va contractar a 75 passatgers voluntaris amb diferents característiques i equipatges de mà de diverses dimensions per tal d'avaluar les diferents estratègies d'embarcament en grups de passatgers heterogenis. La següent figura (6.4) mostra les estratègies d'embarcament avaluades en l'estudi:

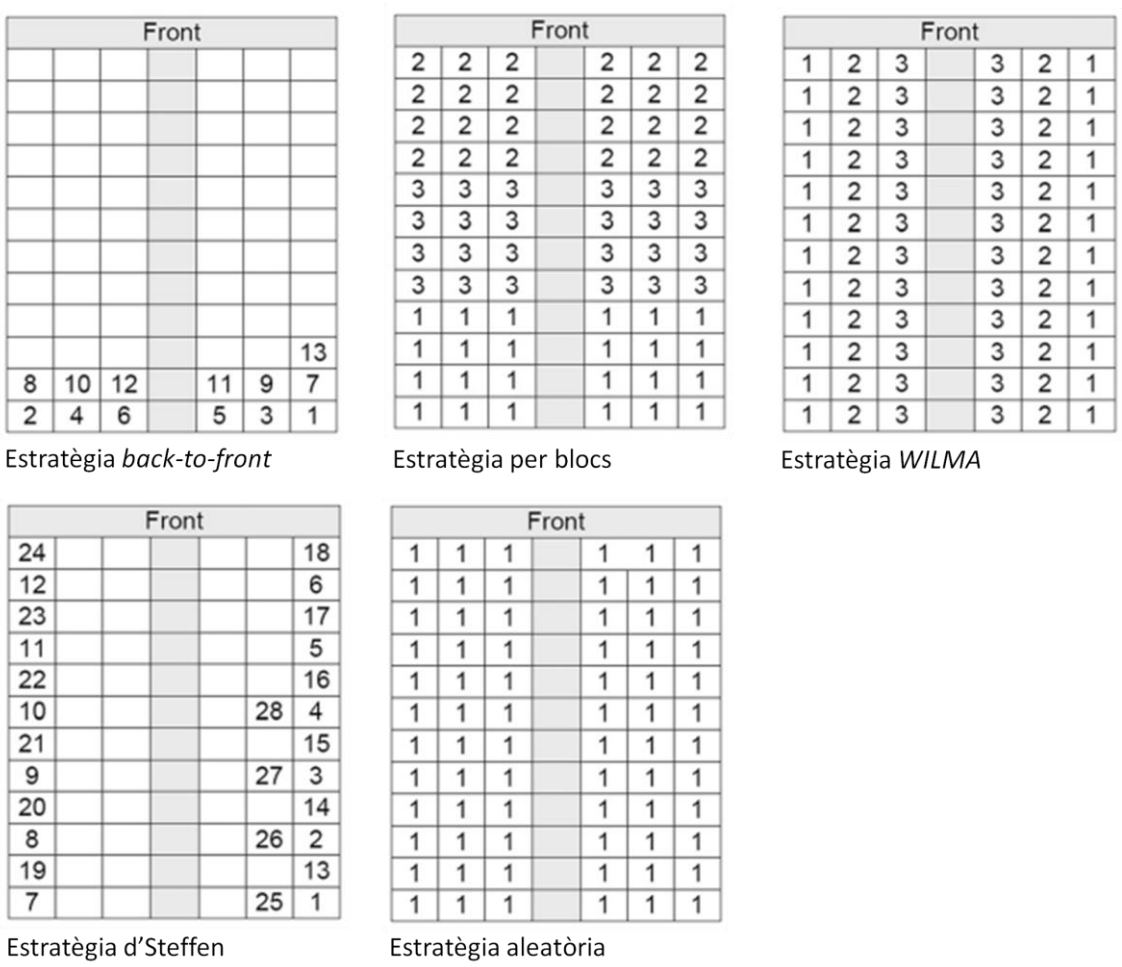


Fig. 6.4 Diagrama de les estratègies d'embarcament avaluades en l'estudi de Steffen i Hotchkiss

Un aclariment important esmentat en l'estudi és que s'observa que les interferències de seient només afecten negativament al temps total d'embarcament quan esdevenen la causa d'interferències al passadís de l'aeronau. Això implica que si, per exemple, s'està embarcant les primeres files de l'avió, interferències de seient causades a les últimes files tindran poc efecte sobre retards en el temps total d'embarcament.

Finalment, tal i com es mostra en la figura 6.5, es comprova que l'estratègia d'embarcament proposada per Steffen resulta ser la més eficient ja que permet distribuir de forma homogènia l'activitat en tota la cabina de l'aeronau, permetent que diversos passatgers desin l'equipatge de mà simultàniament i reduint el nombre d'interferències entre aquests.

Method	Official Time
Back-Front	6:11
Blocks	6:54
Wilma	4:13
Steffen	3:36
Random	4:44

Fig. 6.5 Temps totals d'embarcament per a cada estratègia

En l'estudi realitzat per Tang et al. (2012) es va desenvolupar un model matemàtic per a avaluar diversos mètodes d'embarcament tenint en compte les propietats individuals dels passatgers i les característiques dinàmiques del moviment d'aquests al llarg de tot el procés. Es tracta d'un estudi especialment interessant ja que proposa estratègies d'embarcament no explorades anteriorment pel que resulta oportú realitzar una explicació més detallada al final d'aquest apartat.

A partir de les conclusions extretes en l'estudi de Steffen i Hotchkiss (2011), Cimler et al (2012) avaluaren les mateixes estratègies d'embarcament i en proposaren de noves mitjançant un model de simulació basat en agents múltiples desenvolupat amb NetLogo.

Els autors proposaren una nova estratègia d'embarcament que anomenaren *Kautzka 3*, dissenyada per a minimitzar el nombre d'interferències, possibilitar que diversos passatgers carreguin l'equipatge de mà simultàniament i permetre que famílies i grups reduïts embarquin alhora. En la següent figura (6.6) es mostra una comparativa entre l'estratègia d'embarcament proposada per Steffen (esquerra) i l'estratègia *Kautzka 3* (dreta):

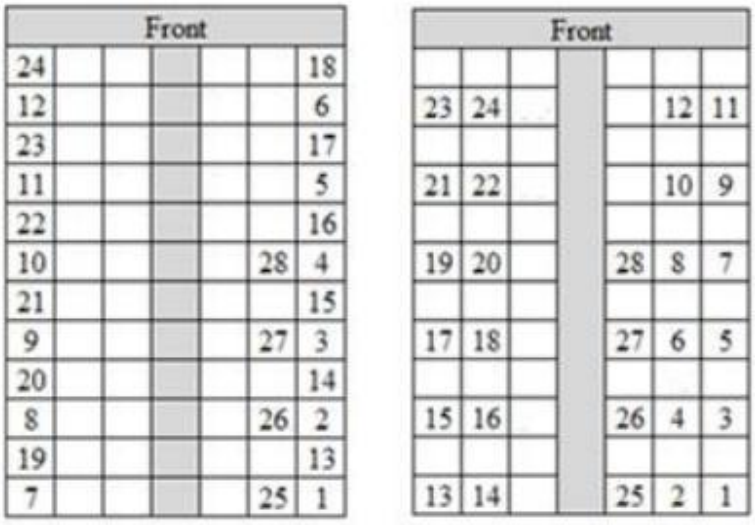


Fig. 6.6 Seqüència d'embarcament de Steffen (esquerra) i *Kautzka 3* (dreta)

A banda de determinar quines de les estratègies esmentades anteriorment són més eficients, l'estudi tenia com a objectiu verificar les següents hipòtesis:

1- Hi ha diferències poc significatives entre el mètode d'embarcament aleatori i el mètode *WILMA*.

2- L'embarcament per blocs és més lent que l'embarcament mitjançant les estratègies *WILMA, Random* i el mètode de Steffen.

Un cop realitzades les simulacions per a cada estratègia d'embarcament i escenari, els autors verificaren que ambdues hipòtesis són certes i estimaren els temps mitjos d'embarcament per a cada estratègia, tal i com es mostra en la següent figura (6.7):

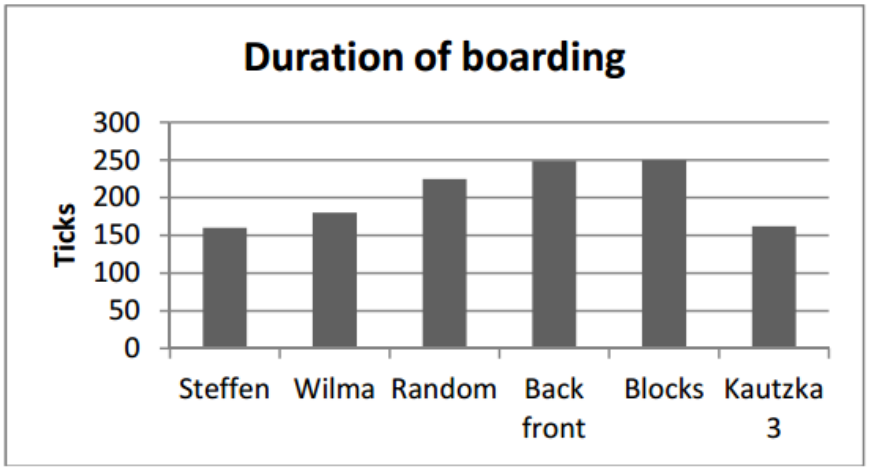


Fig. 6.7 Temps total d'embarcament estimat per a cada estratègia

Per acabar, un apunt important és que també es va mesurar com afectava el retard de passatgers en l'eficiència de cada estratègia d'embarcament. Es va observar que els mètodes de *Kautzka 3* i Steffen eren molt sensibles als nivells de puntualitat, de manera que a partir d'un nivell de passatgers amb retard del 10%, el rendiment d'aquestes estratègies es veia greument afectat i s'assimilava, cada cop més, al rendiment d'un embarcament aleatori.

6.1 Estudis d'especial interès

A continuació s'expliquen amb més detall alguns dels estudis que es consideren d'especial interès per al projecte:

David C. Nyquist (2008): "A study of the airline boarding problem":

L'estudi realitzat per Nyquist i McFadden té per objectiu estudiar el problema d'embarcament d'aeronaus per tal de determinar quina és l'estratègia d'embarcament més eficient que permeti embarcar una aeronau amb un temps mínim sense que això impliqui perjudicar la satisfacció dels passatgers ni la qualitat del servei.

En primer lloc Nyquist i McFadden resumiren estudis realitzats anteriorment per tal de prendre consciència sobre els diferents enfocaments utilitzats per a afrontar el problema així com les conclusions a les quals s'ha arribat en cada un dels estudis en qüestió. Concretament, s'investiguen quatre estudis principals que es resumeixen en la següent taula (fig. 6.8):

Major studies	Authors	Journal	Method	Model	Major findings
Ghent University Study	Van Landeghem and Beuselinck (2000)	European Journal of Operational Research	Simulation	Bin occupancy model	• Best overall strategy: by seat • Best practical strategy: outside-in (seat group) • Average turn time: 30–60 min • Carry-on luggage causes the most congestion
Arizona State University Study	Van den Briel et al. (2005)	Interfaces	Binary integer programming (nonlinear assignment model)	Interference model	• Best strategy: outside-in and reverse-pyramid • Average turn time: 22.9 min • Optimal number of boarding zones: 4 • Two ticket agents: 39% time savings
Institute for Land and Sea Transport Systems Study	Ferrari and Nagel (2005)	Transportation Research Record	Computer simulation sensitivity analysis	The passenger model Average worst case boarding time model	• Best strategy: outside-in or by seat • Those boarding strategies that performed the best under optimal conditions also performed the best under the worst conditions
Boeing Corporation Study	Marelli et al. (1998)	AERO Magazine	Discrete event simulation	PEDS model	• Best strategy: outside-in • Boarding with 2 doors saved 5 min • Boarding with two doors using outside- in saved 17 min

Fig. 6.8 Resum d'estudis anteriors sobre el problema d'embarcament de passatgers en aeronaus

A continuació es realitza un anàlisi per tal de conèixer el temps mig associat a cada estratègia d'embarcament amb l'objectiu d'estimar el impacte financer que això té sobre les companyes aèries. A través de diverses entrevistes realitzades a diferents responsables de US Airways, es determina que el fet d'emprar estratègies d'embarcament tradicionals (per grups o per files) resulta en un temps d'embarcament mig de 30,33 minuts. Per contra, els estudis revisats anteriorment determinaren que el fet d'utilitzar metodologies d'embarcament no tradicionals permet reduir el temps total d'embarcament en 10 minuts; el que implica una reducció del 33%. A més, Nyquist i McFadden utilitzaren el model desenvolupat per Van Landeghem i Beuselinck l'any 2000 per tal d'estimar el temps atribuït a dipositar l'equipatge de mà en els compartiments de la cabina. No obstant, el model en qüestió només tenia en compte el nombre de maletes que cal dipositar i el nombre de maletes que ja hi ha dipositades en el compartiment, ignorant el conflicte que causa el fet que un compartiment estigui

ple i que, com a conseqüència, un passatger hagi de desplaçar-se per tal de trobar un altre compartiment on poder dipositar l'equipatge. Al incorporar aquesta restricció sobre el model s'estima que si l'aerolínia permet embarcar, com a màxim, un equipatge de mà per passatger i, a més, utilitza un mètode d'embarcament no tradicional; és possible reduir el temps d'embarcament en 4,6 minuts. Altrament, si l'aerolínia no permet embarcar cap tipus d'equipatge de mà, és possible reduir el temps total d'embarcament en 11,6 minuts.

A partir de les dades obtingudes anteriorment i assumint un cost per minut a terra de 30$, s'estimen els costos anuals associats a cada estratègia d'embarcament. Per fer-ho, es té en compte el temps mig estimat per a cada estratègia, el cost que implica que una aeronau estigui 1 minut a terra i el nombre de vols diaris. A continuació es mostren les dades estimades en l'estudi (fig. 6.9):

Boarding strategies	Average boarding time (min) (B)	Annual cost (C)	% Cost savings over the traditional method
Traditional	30.33	$498,170,250	–
Non-traditional	19.78	$324,886,500	35
Non-traditional and only one carry-on	15.18	$249,331,500	50
Non-traditional and no carry-on	8.18	$134,356,500	73
Non-traditional, two doors and two carry-on	14.78	$242,761,500	51
Non-traditional, two doors and one carry-on	10.18	$67,206,500	66
Non-traditional, two doors and no carry-on	3.18	$52,231,500	90

[a] Assumes aircraft capacity is at 100% and flights operate 365 days per year.

Fig. 6.9 Cost anual segons diferents estratègies d'embarcament i estalvi respecte a estratègies tradicionals

En conjunt, es conclou que el mètode òptim d'embarcament de passatgers en aeronaus consisteix en utilitzar una estratègia d'embarcament no tradicional, particularment l'anomenada *Reverse pyramid*, que aprofita els beneficis de diverses estratègies tradicionals i maximitza l'espai utilitzat en l'aeronau. A més, l'estudi demostra que les estratègies d'embarcament tradicionals són menys eficients ja que causen un elevat nombre d'interferències i fan un ús poc òptim de l'espai disponible a la cabina de l'aeronau. Així mateix, s'observa que una política d'embarcament que permeti tan sols una peça com a equipatge de mà i embarcar l'aeronau utilitzant més d'una porta pot suposar un estalvi significatiu en el temps total d'embarcament.

Finalment, els autors posen èmfasi en el fet que utilitzar una estratègia d'embarcament eficient té un impacte positiu en la qualitat percebuda pels passatgers i la satisfacció d'aquests donat que són més ràpides, estan millor estructurades i minimitzen el nombre de conflictes en la cabina de l'aeronau.

Jason H. Steffen (2008): "Optimal boarding method for airline passengers":

L'any 2008, Steffen va realitzar un estudi amb l'objectiu de determinar l'ordre en què els passatgers han d'embarcar una aeronau de 20 files i 6 seients per fila per tal que el temps total sigui mínim. A diferència dels estudis realitzats anteriorment, en el model desenvolupat per Steffen s'assumeix que l'activitat que més contribueix en el temps d'embarcament és la de dipositar l'equipatge de mà, mentre que altres tipus de conflicte (per exemple, el fet que un passatger hagi d'aixecar-se per a permetre que un altre passatger segui) no es tenen en compte en el model. Segons l'autor, el fet d'ometre aquests conflictes no afecta greument als resultats de la simulació ja que no suposen la causa principal de temps d'embarcament poc eficients.

Per tal d'assolir els objectius descrits anteriorment i determinar l'ordre en què han d'embarcar els passatgers, s'utilitza l'algoritme d'optimització MCMC ("Markov Chain Monte Carlo"), el qual està constituït pels passos esmentats a la figura 6.10.

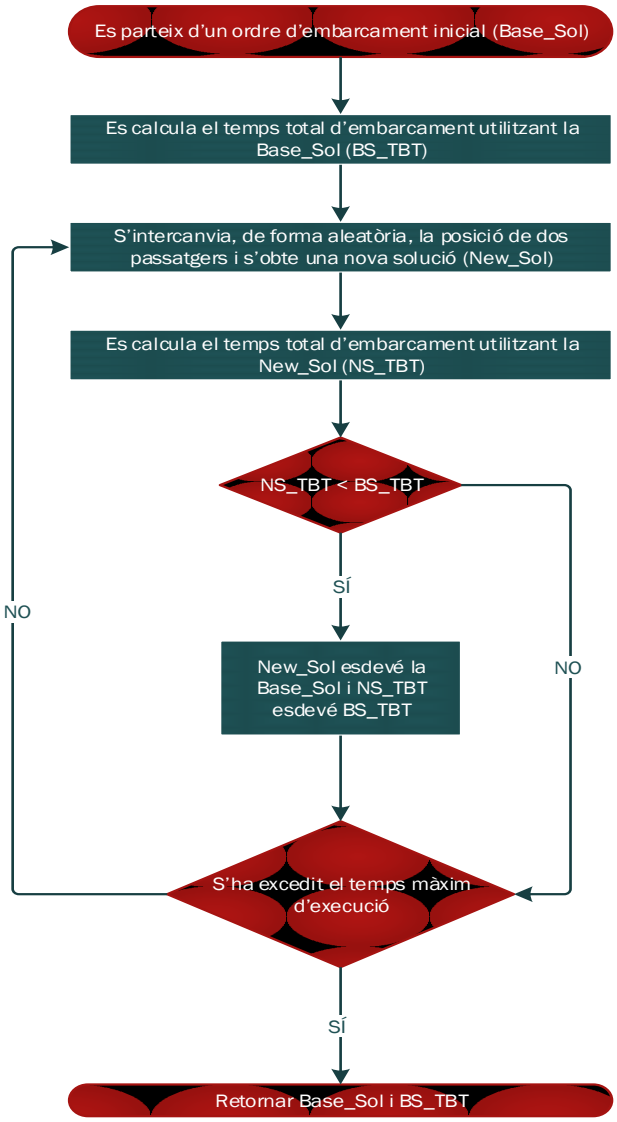


Fig. 6.10 Algoritme MCMC

L'algoritme s'executa, aproximadament, 10.000 iteracions ja que el fet d'afegir-ne més no millora els resultats de forma significativa.

Els resultats obtinguts mostren que hi ha vàries configuracions diferents que resulten en temps totals d'embarcament molt similars, pel que es demostra que no existeix una única configuració òptima sinó que diverses configuracions poden ser equivalents. Per contra, el temps individual que un passatger triga en carregar l'equipatge de mà té un efecte significatiu en el temps total d'embarcament. S'observa que com més elevat és el temps que triga un passatger en carregar l'equipatge de mà, menys important és l'ordre en què els passatgers embarquen. Això implica que l'ordre que minimitza el temps total d'embarcament està directament relacionat amb el nombre de passatgers que poden carregar l'equipatge de mà de forma simultània. Els resultats obtinguts permeten determinar que els passatgers necessiten una separació mínima d'una fila per a poder carregar els seus respectius equipatges simultàniament. D'aquesta manera Steffen determina que una estratègia d'embarcament òptima pot ser aquella que permet maximitzar el nombre de passatgers que carreguen l'equipatge de forma simultània, començant pels seients de les finestres i les últimes files de l'avió. A continuació es mostra un exemple de l'ordre d'embarcament òptim (fig. 6.11):

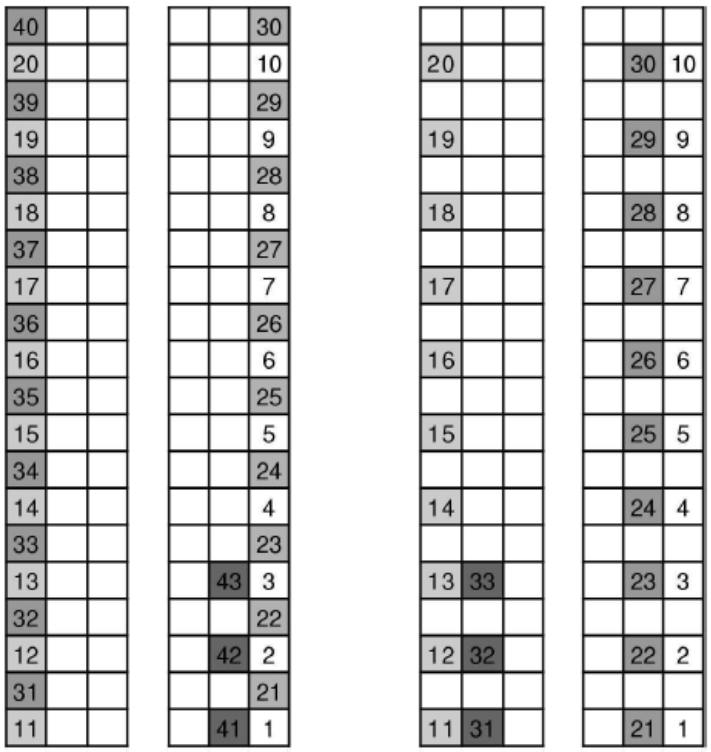


Fig. 6.11 Seqüència d'embarcament establerta en l'estratègia d'embarcament proposada per Steffen

Finalment, s'avalua la robustesa de l'estratègia d'embarcament proposada mitjançant la realització de dos experiments diferents. En primer lloc, s'avalua el temps total d'embarcament utilitzant diferents distribucions per al temps de càrrega de

l'equipatge de mà. S'observa que el tipus de distribució no té un impacte significatiu sobre el temps total d'embarcament i que, el fet que un passatger trigui en carregar l'equipatge un temps considerablement més elevat que la resta de passatgers no afecta de forma significativa en l'eficiència de l'estratègia d'embarcament. En segon lloc, es realitzen canvis aleatoris sobre l'ordre d'embarcament dels passatgers consistents en intercanviar l'ordre de diverses parelles de passatgers. En aquest cas s'observa que el fet de realitzar canvis aleatoris sobre la posició dels passatgers té un efecte significatiu en el temps total d'embarcament, de tal manera que si s'intercanvia un 10% dels passatgers, el temps d'embarcament pot patir un increment de fins al 20%. Amb tot, al arribar a un cert punt d'aleatorietat, el temps total d'embarcament s'estanca i deixa de créixer.

Es conclou que utilitzar l'estratègia d'embarcament *back-to-front* és molt poc eficient ja que deixa molt d'espai disponible sense utilitzar, pel que resulta molt més eficient distribuir els passatgers al llarg del passadís de l'aeronau de manera que puguin embarcar simultàniament. A més, aconseguir reduir el temps total d'embarcament té un efecte doblement positiu ja que beneficia tant a la companyia aèria com als seus clients.

Tie-Quiao Tang (2012): "An aircraft boarding model accounting for passengers individual properties":

En aquest estudi realitzat l'any 2012 s'afronta el problema d'optimització del procés d'embarcament d'aeronaus basant-se, sobretot, en les característiques individuals dels passatgers així com les característiques del seu moviment durant tot el procés d'embarcament (des de la porta d'embarcament fins als seients ocupats pels passatgers).

En primera instància, es realitza una revisió de diversos estudis relacionats amb el problema d'embarcament de passatgers en aeronaus on es conclou que gran part dels estudis realitzats fins al moment es basen en les restriccions físiques de la cabina de l'aeronau, sense tenir en compte variacions en el moviment dels passatgers al llarg de tot el procés i sense considerar les propietats individuals d'aquests. Un cop identificada la motivació principal de l'estudi, es proposa un model matemàtic basat en la teoria de flux de vianants per tal de definir matemàticament l'equació que descriu el moviment dels passatgers al llarg de tot el procés. A més, el model també té en compte el impacte que tenen les propietats individuals dels passatgers (l'equipatge de mà, per exemple) sobre el moviment del conjunt de passatgers i l'efecte sobre el temps total d'embarcament mitjançant la imposició de restriccions matemàtiques sobre l'equació de moviment de cada passatger. Això permet que el model consideri que les característiques de cada passatger condicionen les propietats de la resta, per exemple, la seva velocitat màxima, la distància entre passatgers, el retràs causat al dipositar l'equipatge, etc...

A continuació, se sotmet el model a una sèrie de tests numèrics per tal de provar que aquest és capaç de descriure de forma qualitativa les propietats dinàmiques del moviment dels passatgers sota tres estratègies d'embarcament diferents: aleatòria, amb seients pre-assignats (*back-to-front*) i amb assignació de seients segons les propietats individuals dels passatgers. En el cas de les dues primeres estratègies d'embarcament s'observa que ambdues ocasionen avançaments entre els passatgers, congestions i conflictes de seient, pel que és necessari dissenyar una estratègia que tingui en compte tant l'assignació de seients com les característiques individuals dels passatgers. Per a fer-ho, es divideix la cabina en tres seccions diferents i es realitza una assignació de seients d'acord a la velocitat de desplaçament òptima de cada passatger i el nombre de peces d'equipatge de mà. Això permet que, dintre de cada secció, si $n > m$, l'efecte que té l'equipatge de mà sobre la velocitat de moviment del passatger $n_{è}$ és menor que l'efecte que pateix sobre el passatger $m_{è}$ i que la velocitat màxima i òptima del passatger $n_{è}$ és major que la del passatger $m_{è}$.

En la següent taula (fig. 6.12) es mostra un resum de cada estratègia d'embarcament estudiada en el model:

	The main properties	The main parameters	The main results
I	The passengers randomly board the aircraft	M_n, α_n, $\lambda_{1,n}$, $\lambda_{2,n}$, $T_{1,n}$, $T_{2,n}^{\text{lugg}}$, $T_{2,n}^{\text{con}}$, $v_n^{\max}$, h_n^c are independent variables with randomness; ε_n, η_n, $\theta_{1,n}$, $\theta_{2,n}$, ω_n are constant	Congestion, jam, overtaking, queue-jumping, seat conflict and waste time occur
II	The passengers board the aircraft based on their seat serial number	$T_{1,0} = 0$ and $T_{1,n,s}$ is a random integer from 1 to 2; other parameters are the same as those of the aircraft boarding strategy (I)	The results are similar but not as severe as those obtained by the aircraft boarding strategy (I)
III	The passengers are divided into three groups; in each group, the passengers board the aircraft based on their seat serial numbers that are determined by their individual properties; each passenger's ticket is automatically checked at the gate desk	$T_{1,0} = T_{1,n,s} = 0$; in each group, if $n < m$, then $V_n(\bullet) > V_m(\bullet)$ $v_n^{\max} > v_m^{\max}$; other parameters are the $M_n < M_m$ same as those of the aircraft boarding strategy (I)	Congestion, jam, overtaking, queue-jumping, seat conflict and waste time do not occur; each passenger can board the aircraft approximately at his/her maximum speed; the aircraft boarding time is less than those of the other two aircraft boarding strategies

Fig. 6.12 Taula resum de les diferents estratègies d'embarcament avaluades en l'estudi

Finalment, es conclou que l'estratègia basada en l'assignació de seients en funció de les característiques individuals de cada passatger és la que dóna millors resultats ja que permet eliminar la formació de cues i avançaments així com minimitzar el nombre de conflictes ocasionats per passatgers que han de carregar el seu equipatge i passatgers que han d'aixecar-se per tal de permetre que altres passatgers puguin ocupar els seus respectius seients.

Secció 3: Algorismes *ABP_SolGen* i *ABP_RTSA*

7. Estratègia d'embarcament *ABP_SolGen*

En el següent apartat s'explica de forma detallada la seqüència de passos que formen l'estratègia d'embarcament *ABP_SolGen* ("*Aircraft Boarding Problem Solution Generator*"), utilitzada per a obtenir una solució al problema d'embarcament de passatgers en aeronaus tenint en compte les característiques individuals dels passatgers així com les relacions existents entre aquests.

L'estratègia té per objectiu proporcionar solucions el més eficient possibles tenint en compte la seva capacitat per a ser implementades en l'operativa diària d'una companyia aèria. A diferència d'altres estratègies d'embarcament teòriques proposades en estudis anteriors com, per exemple, l'estratègia *WILMA* o *Kautzka 3*, l'estratègia *ABP_SolGen* permet realitzar una assignació de seients a passatgers i definir l'ordre en què aquests embarcaran l'aeronau mitjançant la creació de grups d'embarcament tenint en compte que, en cap cas, és possible separar els passatgers que viatgen conjuntament ni tampoc és possible ordenar els passatgers a la cua d'embarcament.

D'alguna manera es podria considerar l'estratègia *ABP_SolGen* com un híbrid d'altres estratègies d'embarcament en el sentit que pretén beneficiar-se de les fortaleses de cada estratègia però sempre tenint en compte les restriccions exposades anteriorment per tal d'obtenir solucions eficients, pràctiques i realistes.

A continuació es mostra un diagrama (fig. 7.1) en què es resumeixen els diversos passos que constitueixen l'estratègia d'embarcament *ABP_SolGen*, a través dels quals és possible obtenir una solució eficient i realista al problema d'embarcament de passatgers en aeronaus. D'altra banda, en el següent apartat s'explica, de forma detallada, cadascun d'aquests passos.

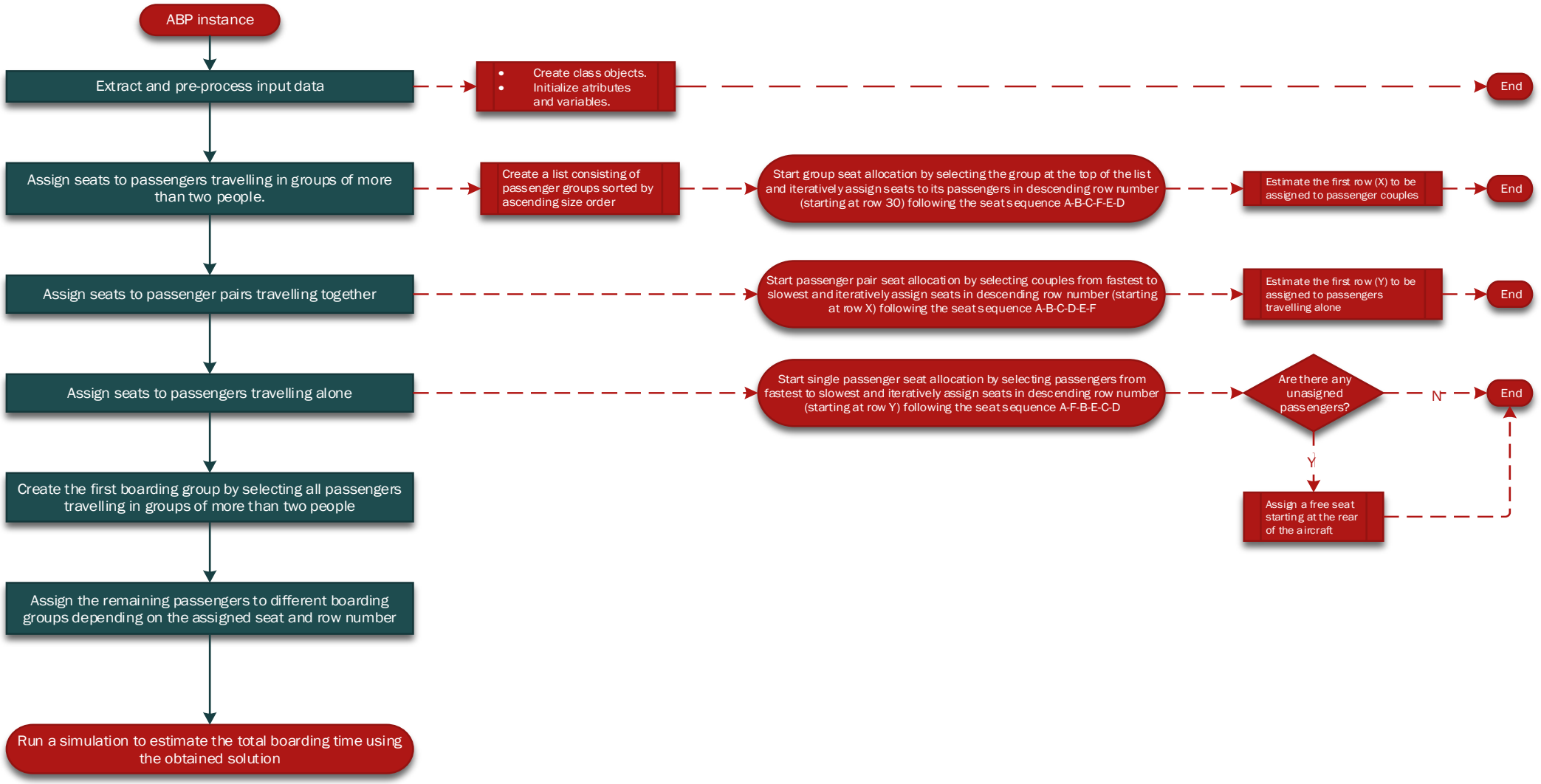


Fig. 7.1 Diagrama dels passos utilitzats per a obtenir una solució mitjançant l'estratègia d'embarcament *ABP_SolGen*

7.1 Funcionament de l'estratègia

1. Extracció d'inputs i pre-processament dels passatgers:

1.1 – Extracció de les dades relatives als passatgers d'un vol determinat (inputs del problema), les quals es troben emmagatzemades en un fitxer .txt que conté el codi de reserva (localitzador del PNR[3]) de cada passatger així com l'any de naixement d'aquest, tal i com es mostra en el següent exemple (fig. 7.2):

#book_ref	year
23Z7ZB	1972
78G6BB	1963
43F5FJ	1927
43F5FJ	1998
43F5FJ	1991
98L6CN	1980
74S5DD	1965
74S5DD	1968
11F9QA	1988
11F9QA	1988
22F5JK	1957
13L4PX	1960
88G9GF	1985
93H9JC	1992
74K3LM	2000
46V1DE	1972
77H4SL	1974
77H4SL	1995
56B5XS	1978
91J2AP	1930
32H8BA	1981

Fig. 7.2 Exemple del fitxer d'inputs .txt

1.2 - Creació d'una llista d'objectes de la classe passatger per a cada línia introduïda al fitxer d'inputs. A més, per a cada passatger, s'inicialitzen alguns dels atributs en funció del codi de reserva i l'any de naixement com, per exemple, el tipus de passatger, l'edat o la velocitat a la que es desplaçarà durant l'embarcament.

1.3 - Creació de grups de passatgers que viatgen conjuntament a partir del codi de reserva. Tots els passatgers amb codis de reserva iguals

[3] PNR: De l'anglès *Passenger Name Record,* fa referència a tota la informació enregistrada al sistema de reserves en relació a la reserva d'un passatger o grup de passatgers. El localitzador és un codi alfanumèric de sis xifres que permet identificar, de forma única, cada PNR.

pertanyen al mateix PNR, pel que s'assumeix que viatgen junts. D'aquesta manera, en l'exemple anterior hi hauria un grup format per tres passatgers (amb el codi 43F5FJ), tres grups formats per dos passatgers i dotze grups formats per un únic passatger.

1.4 - Creació d'una llista formada per diversos grups d'embarcament en funció de la dimensió de cada grup. Cada grup de passatgers s'assigna a una de tres llistes possibles en funció del nombre d'integrants del grup (un, dos o més de dos).

2. Assignació de seients a passatgers:

A diferència de moltes estratègies d'embarcament i d'assignació de seients a passatgers proposades en altres estudis, l'algoritme *ABP_SolGen* permet realitzar una assignació basada en les característiques dels passatgers així com les relacions existents entre aquests. Això permet obtenir solucions òptimes i implementables, ja que en cap cas se separa a passatgers que viatgen junts durant l'embarcament.

És important remarcar que en l'assignació de seients no s'estableix l'ordre en què els passatgers embarcaran l'aeronau, fet que s'aconseguirà mitjançant l'assignació de passatgers a grups d'embarcament. No obstant, l'assignació de seients a passatgers resulta extremadament important ja que permet no separar a passatgers que viatgen junts (parelles, grups i famílies) i tenir en compte la distància que haurà de recórrer cada passatger per arribar al seu seient, cosa que permetrà realitzar una assignació que permeti minimitzar el temps total d'embarcament.

2.1 - Assignació de seients a grups formats per més de dos passatgers:

2.1.1- Ordenació de la llista de grups de més de dos passatgers per ordre de dimensió ascendent. D'aquesta manera s'aconsegueix assignar primer els grups formats per tres passatgers per tal de que puguin ocupar mitja filera sencera sense ser separats.

2.1.2- Assignació de seients a passatgers en ordre de dimensió creixent i ordre de fila decreixent a partir de la última fila de l'aeronau i d'acord amb la seqüència d'assignació de seients A-B-C-F-E-D per tal d'evitar qualsevol tipus de conflicte de seient, tal i com es mostra en el següent exemple (fig. 7.3):

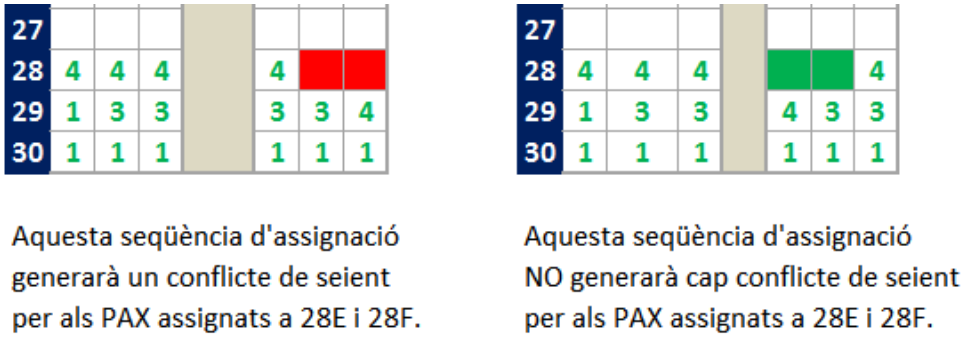


Fig. 7.3 Exemple d'assignació de seients a grups de més de dos passatgers

2.1.2- Càlcul de la primera fila (X) a assignar a passatgers que viatgen en parelles. A continuació s'explica un exemple:

Suposem que en un vol determinat viatgen un total de 23 passatgers en grups formats per més de 2 membres. En aquest cas, 23/6 = 3,83; pel que es considera que caldrà ocupar 4 files per a acomodar a aquests passatgers. Tenint aquest fet en compte i suposant que l'aeronau té un total de 30 files, l'assignació de passatgers que viatgin en parelles es realitzarà a partir de la fila 26.

2.2- Assignació de seients a parelles:

2.2.1- Ordenació de la llista de grups formats per parelles de passatgers en funció de la tipologia dels passatgers compresos en cada grup.

Si es distingeix entre dos tipus de passatgers diferents en funció de la seva velocitat de moviment (*Standard* i *Slow*), la llista estarà encapçalada per grups formats per *Standard+Standard*, a continuació hi haurà els grups formats per *Standard+Slow* i, finalment, aquells que estiguin formats per *Slow+Slow.*

2.2.2- Assignació de seients a parelles de passatgers tenint en compte la tipologia dels passatgers integrants de cada grup (d'acord amb l'ordre establert a la llista) i en ordre de fila decreixent (a partir de la fila X) segons la seqüència d'assignació de seients A-B-C-D-E-F.

En aquest cas, s'assignarà seients de les files més llunyanes als passatgers amb una velocitat de moviment més elevada per tal de que els passatgers més lents hagin de recórrer el mínim de distància possible per a ocupar el seu seient i, d'aquesta manera, minimitzar el temps total d'embarcament. D'altra banda, donat que el nombre de seients per fila és parell, la seqüència d'assignació permet no separar a cap parella.

La següent figura mostra un exemple de l'assignació de seients a passatgers d'acord amb el criteri descrit anteriorment. El color verd indica passatgers del tipus *Slow,* mentre que el vermell fa referència als passatgers del tipus *Standard:*

15	39	39	x		x	x	x
16	36	36	37		37	38	38
17	33	33	34		34	35	35
18	30	30	31		31	32	32
19	27	27	28		28	29	29
20	24	24	25		25	26	26
21	21	21	22		22	23	23
22	18	18	19		19	20	20
23	15	15	16		16	17	17
24	12	12	13		13	14	14
25	9	9	10		10	11	11
26	6	6	7		7	8	8

Fig. 7.4 Exemple d'assignació de seients a parelles de passatgers

2.2.3- Càlcul de la primera fila (Y) a assignar a passatgers que viatgen sols en funció del nombre de files ocupades pels passatgers que viatgen en parelles.

2.3- Assignació de seients a passatgers individuals:

2.3.1- Ordenació de la llista de passatgers individuals en funció de la tipologia de cada passatger. En aquest cas, es començarà assignant seients

als passatgers del tipus *Slow* per tal de minimitzar l'efecte de possibles conflictes de seient, tal i com s'explica en els següents punts.

2.3.2- Assignació de seients a passatgers individuals tenint en compte la tipologia dels passatgers i en ordre de fila decreixent (a partir de la fila Y), d'acord amb la seqüència d'assignació de seients A-F-B-E-C-D tal i com es mostra a continuació (fig. 7.5):

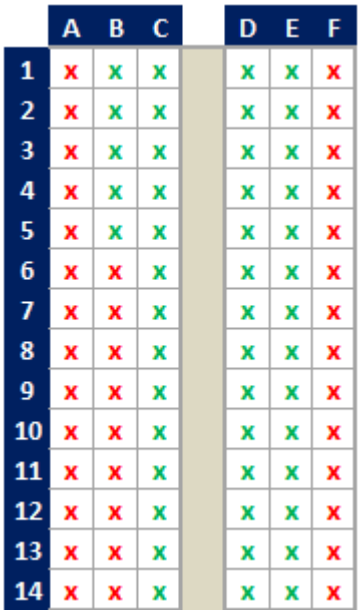


Fig. 7.5 Exemple d'assignació de seients a passatgers individuals

En aquest cas, els passatgers del tipus *Slow* es representen en color vermell mentre que els passatgers *Standard* es representen en color verd.

La seqüència d'assignació detallada anteriorment permetrà minimitzar l'efecte que tindran els conflictes de seient sobre el temps total d'embarcament. Això és així perquè es començarà assignant els seients situats a la finestra (A i F) a tots els passatgers del tipus *Slow* per tal de maximitzar la probabilitat de que, en cas d'haver-hi un conflicte de seient, el passatger que hagi d'aixecar-se per a cedir el pas sigui del tipus *Standard.*

En la figura 7.6 es mostra una situació en què hi ha un conflicte de seient ja que el passatger 2 ha d'aixecar-se per tal de permetre que el passatger 3

ocupi el seient al qual ha estat assignat (A1). D'acord amb la seqüència d'assignació descrita anteriorment, la probabilitat de que el passatger 2 sigui del tipus *Slow* és més baixa que si es segueix alguna altra seqüència d'assignació, cosa que permetrà resoldre el conflicte de forma més ràpida i minimitzar la repercussió sobre el temps total d'embarcament.

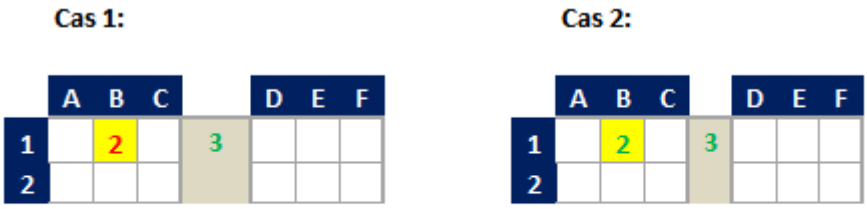


Fig. 7.6 Exemple d'un possible conflicte de seient

2.3.3- En cas de no haver acomodat a tots els passatgers individuals, es procedirà a assignar aquells seients que hagin quedat lliures als passatgers restants en ordre de fila decreixent.

La següent figura (7.7) mostra un exemple d'una assignació de seients a passatgers seguint els passos descrits anteriorment. En aquest cas hi ha un total de 5 grups formats per més de 3 passatgers, 33 parelles i 92 passatgers individuals. Els números de cada cel·la representen els identificadors únics de cada grup o parella, mentre que el color de cada número representa el tipus de passatger (verd=*Standard* i vermell=*Slow*). Les "x" representen cada passatger individual. És important remarcar que en aquesta figura només es mostra l'assignació inicial de seients, sense tenir en compte els grups d'embarcament ni l'ordre en què aquests grups embarcaran l'aeronau.

	A	B	C		D	E	F
1	x	x	x		x	x	x
2	x	x	x		x	x	x
3	x	x	x		x	x	x
4	x	x	x		x	x	x
5	x	x	x		x	x	x
6	x	x	x		x	x	x
7	x	x	x		x	x	x
8	x	x	x		x	x	x
9	x	x	x		x	x	x
10	x	x	x		x	x	x
11	x	x	x		x	x	x
12	x	x	x		x	x	x
13	x	x	x		x	x	x
14	x	x	x		x	x	x
15	39	39	x	PASSADÍS	x	x	x
16	36	36	37		37	38	38
17	33	33	34		34	35	35
18	30	30	31		31	32	32
19	27	27	28		28	29	29
20	24	24	25		25	26	26
21	21	21	22		22	23	23
22	18	18	19		19	20	20
23	15	15	16		16	17	17
24	12	12	13		13	14	14
25	9	9	10		10	11	11
26	6	6	7		7	8	8
27	5	5	5		x	x	5
28	4	4	4		5	5	5
29	3	3	3		4	4	3
30	1	1	1		2	2	2

Grups > 2 PAX
Parelles
PAX Individuals

Fig.7.7 Exemple d'una assignació de seients a passatgers mitjançant l'estratègia *ABP_SolGen*

3. Assignació de passatgers a grups d'embarcament:

L'assignació de passatgers a grups d'embarcament resulta d'extremada importància ja que permet definir l'ordre en què els passatgers embarcaran l'aeronau en funció del seient al qual han estat assignats. No obstant, a l'hora de crear els grups d'embarcament és important tenir en compte la seva aplicabilitat en l'operativa real ja que no és possible preveure l'ordre en què els passatgers es disposaran al llarg de la cua a les portes d'embarcament ni cridar de forma individual a cada passatger per a que procedeixi a embarcar l'aeronau, cosa que implicaria separar a grups i famílies durant tot el procés. És per aquest motiu que no seria realista assumir per part de la companyia aèria el control, a nivell individual, de l'ordre en què els passatgers accediran l'aeronau.

Amb tot, la creació de grups d'embarcament permetrà dividir la totalitat dels passatgers en diversos grups per tal de poder establir un ordre d'embarcament basat en l'assignació de seients prèvia que permeti maximitzar el nivell d'activitat dins de la cabina de l'aeronau i minimitzar el nombre d'interferències (conflictes) entre els passatgers durant tot el procés, cosa que permetrà reduir el temps total d'embarcament. Concretament, la política de creació de grups d'embarcament exposada a continuació permetrà eliminar gran part dels conflictes de seient causats entre els passatgers. Per les raons descrites anteriorment no serà possible eliminar els conflictes de passadís, ja que els passatgers no necessàriament accediran a la cabina de l'avió en ordre de fila decreixent.

3.1- Assignació a grups i famílies de més de 2 membres:

Els primers passatgers en embarcar l'aeronau i que formaran el primer grup d'embarcament seran tots aquells que viatgen en grups formats per més de dos passatgers, els quals hauran estat assignats a les darreres files de l'aeronau.

3.2- Assignació a parelles de passatgers:

Els passatgers que viatgin en parelles seran assignats a grups d'embarcament en funció del número de fila i seient al qual hagin estat assignats d'acord amb la política detallada a continuació:

- Els passatgers amb fila parell i seient A o B seran assignats al segon grup d'embarcament.
- Els passatgers amb fila imparell i seient A o B seran assignats al tercer grup d'embarcament.

- Els passatgers amb fila parell i seient E o F seran assignats al quart grup d'embarcament.

- Els passatgers amb fila imparell i seient E o F seran assignats al cinquè grup d'embarcament.

- Els passatgers amb fila parell i seient C o D seran assignats al sisè grup d'embarcament.

- Els passatgers amb fila imparell i seient C o D seran assignats al setè grup d'embarcament.

Aquesta política d'assignació de grups d'embarcament permetrà no separar a les parelles durant l'embarcament, evitar qualsevol tipus de conflicte de seient i deixar suficient espai entre els passatgers per tal que l'embarcament es dugui a terme de forma còmode i fluida.

3.3- Assignació a passatgers individuals:

L'assignació de passatgers individuals a grups d'embarcament tindrà per objectiu evitar qualsevol tipus de conflicte de seient. Conseqüentment, els passatgers individuals seran assignats tal i com s'exposa a continuació:

- Els passatgers amb seient A seran assignats al segon grup d'embarcament.
- Els passatgers amb seient B seran assignats al tercer grup d'embarcament.
- Els passatgers amb seient C seran assignats al sisè grup d'embarcament.
- Els passatgers amb seient D seran assignats al setè grup d'embarcament.
- Els passatgers amb seient E seran assignats al cinquè grup d'embarcament.
- Els passatgers amb seient F seran assignats al quart grup d'embarcament.

Reprenent l'exemple exposat a la figura 7.7 i d'acord amb la política d'assignació de passatgers a grups d'embarcament exposada anteriorment, l'ordre en què els passatgers embarcarien l'aeronau seria el que es mostra a continuació (fig. 7.8):

	A	B	C		D	E	F
1	2	3	6		7	5	4
2	2	3	6		7	5	4
3	2	3	6		7	5	4
4	2	3	6		7	5	4
5	2	3	6		7	5	4
6	2	3	6		7	5	4
7	2	3	6		7	5	4
8	2	3	6		7	5	4
9	2	3	6		7	5	4
10	2	3	6		7	5	4
11	2	3	6		7	5	4
12	2	3	6		7	5	4
13	2	3	6		7	5	4
14	2	3	6		7	5	4
15	3	3	6	PASSADÍS	7	5	4
16	2	2	6		6	4	4
17	3	3	7		7	5	5
18	2	2	6		6	4	4
19	3	3	7		7	5	5
20	2	2	6		6	4	4
21	3	3	7		7	5	5
22	2	2	6		6	4	4
23	3	3	7		7	5	5
24	2	2	6		6	4	4
25	3	3	7		7	5	5
26	2	2	6		6	4	4
27	1	1	1		7	5	1
28	1	1	1		1	1	1
29	1	1	1		1	1	1
30	1	1	1		1	1	1

Grups > 2 PAX

Parelles

PAX Individuals

Fig.7.8 Exemple de l'assignació de passatgers a grups d'embarcament mitjançant l'estratègia *ABP_SolGen*

8. Estratègia d'embarcament *ABP_RTSA*

L'estratègia d'embarcament *ABP_RTSA* ("*Aircraft Boarding Problem Real Time Seat Allocation"*) consisteix en assignar seients als passatgers que es troben a la cua de les portes d'embarcament. D'aquesta manera, s'assigna seients als passatgers a mesura que aquests accedeixen a l'aeronau tenint en compte si viatgen sols o acompanyats per tal de no separar a grups ni famílies.

Partint del supòsit que tota la informació relativa a la reserva de cada passatger es troba emmagatzemada a la targeta d'embarcament, serà possible conèixer l'ordre en què els passatgers s'hauran disposat a la cua de les portes d'embarcament així com quins d'aquests passatgers viatgen conjuntament i han de ser assignats a seients contigus. Es tracta, per tant, d'una aproximació realista i molt innovadora per tal de donar solució al problema d'embarcament de passatgers en aeronaus ja que, fins al moment, no ha estat possible proposar una solució realista basada en l'ordre en què els passatgers accedeixen a l'aeronau.

A diferència de la metodologia emprada en l'estratègia de resolució *ABP_SolGen* en què, més enllà de l'ordre marcat per la definició de grups d'embarcament, l'ordre en què els passatgers accedeixen a l'aeronau és desconegut, fet que impedeix eliminar els possibles conflictes de passadís causats entre els passatgers, l'assignació de seients a temps real permet neutralitzar els possibles conflictes de seient i de passadís mitjançant una adaptació de l'estratègia d'embarcament *WILMA*. A més, un factor que converteix l'assignació de seients a temps real en una estratègia d'embarcament altament robusta i eficient és el fet que no depèn del nivell de compliment i puntualitat dels passatgers envers a la política d'embarcament emprada per la companyia aèria. Aquest fet converteix l'assignació de seients a temps real en una estratègia d'embarcament molt flexible ja que permet proposar solucions eficients i realistes independentment de les condicions inicials del problema.

Per tal de garantir que les solucions proposades per aquesta estratègia d'embarcament siguin el més eficients possible, caldrà tenir en compte els següents factors:

- Minimitzar el nombre total de conflictes de seient mitjançant una assignació de seients des de l'exterior cap a l'interior (des de les finestres cap al passadís).

- Minimitzar els conflictes de passadís causats per passatgers que impedeixen que altres passatgers puguin arribar als seients als que han estat assignats mitjançant una assignació per ordre de fila descendent. Per tant, l'assignació de seients es realitzarà començant per la última fila de l'avió.

- Maximitzar el nivell d'activitat dintre de la cabina de l'avió permetent l'embarcament simultani del màxim nombre de passatgers.

- Realitzar una assignació de seients que permeti mantenir a grups i famílies juntes al llarg de tot el procés.

8.1 Funcionament de l'estratègia

Amb l'objectiu d'obtenir solucions eficients que permetin reduir el temps total d'embarcament sense afectar greument la satisfacció dels passatgers i tenint en compte tots els factors descrits anteriorment, la metodologia que caldrà seguir per tal de realitzar una assignació de seients a temps real serà la que es detalla a continuació:

Abans d'iniciar l'assignació de seients caldrà inicialitzar el valor de la fila actual (*currRow)* i el seient actual (*currSeat*). Tenint en compte que l'assignació de seients es realitzarà en ordre de fila decreixent i d'acord amb la seqüència d'assignació de seients A-B-F-E-D-C (1-2-6-5-4-3) per tal d'evitar qualsevol tipus de conflicte de seient i maximitzar la probabilitat de no separar a famílies, el valor de *currRow* serà 30 i el de *currSeat* serà 1. Tal i com es mostra en la figura 8.1, la seqüència d'assignació de seients A-B-F-E-D-C, a diferència de la seqüència de l'estratègia *WILMA* (A-B-C-F-E-D), permet maximitzar la probabilitat de mantenir a parelles de passatgers juntes al inici de l'assignació .

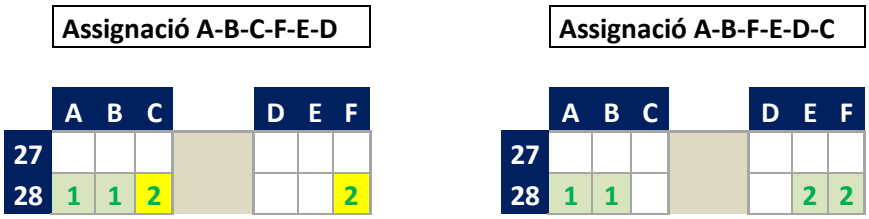


Fig. 8.1 Diferències respecte diferents seqüències d'assignació de seients

Si bé és cert que hom podria argumentar que, de la mateixa manera que l'assignació de seients A-B-F-E-D-C permet maximitzar la probabilitat de mantenir a les parelles juntes, la seqüència A-B-C-F-E-D maximitza la probabilitat de no separar a passatgers que viatgen en grups de tres persones, es considera que l'assignació A-B-F-E-D-C és més beneficiosa perquè el impacte de separar a parelles és més negatiu.

Un cop inicialitzat el valor de *currRow* i *currSeat*, es procedirà a assignar seients als passatgers seguint els següents passos:

0- Mentre hi hagi passatgers a la cua i la fila assignada al passatger anterior (*prevRow*) sigui major que 1, caldrà seguir els següents passos:

1- Seleccionar el primer passatger de la cua (*currPAX*) per tal d'assignar-li un seient.

2- Comparar el codi de reserva del passatger actual (*currPAX*) amb el del passatger anterior (*prevPAX*) per tal de determinar si els passatgers viatgen conjuntament i, conseqüentment, han de ser assignats a seients contigus.

   a. Si el codi de reserva del *currPAX* coincideix amb el del *prevPAX*, llavors caldrà comprovar que hi hagi seients lliures a la mateixa fila del *prevPAX* (*prevRow*). Tenint en compte que l'assignació de seients es realitzarà d'acord amb la seqüència A-B-F-E-D-C (1-2-6-5-4-3), si *prevSeat*=4 el *prevPAX* haurà ocupat l'últim seient de la *prevRow*, de manera que *currRow* haurà de ser *prevRow* - 1 i *currSeat* serà el primer seient lliure de la *currRow* d'acord amb la seqüència. En cas contrari, el valor de *currRow* serà igual que el de *prevRow* i el *currSeat* serà el seient següent al *prevSeat* d'acord amb la seqüència d'assignació de seients.

   b. Si el codi de reserva del *currPAX* no coincideix amb el del *prevPAX* s'assumirà que el passatger actual no viatja amb el passatger anterior, pel que no serà necessari assignar-li un seient contigu. Tenint aquest fet en compte i amb l'objectiu de maximitzar el nombre de passatgers que realitzen l'embarcament de forma simultània sense que interfereixin entre sí, la fila que s'assignarà al passatger actual (*currRow*) serà *prevRow* – n, sent n[4] el nombre de passatgers del grup anterior. D'altra banda, *currSeat* serà el primer seient lliure de la *currRow* d'acord amb la seqüència d'assignació de seients esmentada anteriorment.

En ambdós casos, en cas que no hi hagués cap seient lliure a *currRow*, caldria passar a la següent fila (*currRow* = *currRow* – 1) i comprovar si hi ha algun seient lliure. Aquest procés s'hauria de repetir fins a trobar una fila amb algun seient lliure.

És important remarcar que el fet de mantenir una distància de separació entre les files de passatgers que no viatgen conjuntament resulta en un estalvi molt elevat del temps total d'embarcament ja que permet minimitzar el nivell d'ociositat dintre de la cabina

[4] Serà possible obtenir el valor d'n comptant el nombre de passatgers amb codi de reserva igual al del *prevPAX* (incloent el *prevPAX*).

de l'aeronau. En la següent figura (8.2) es mostra un exemple en què, en el cas 1, els passatgers pertanyents al grup 2 han estat assignats a la mateixa fila que els passatgers

del grup 1 de manera que han d'esperar que tots els passatgers ocupin els seus respectius seients per a poder arribar a la fila a la qual han estat assignats. Per contra, en el cas 2 s'han assignat files diferents per a cada grup de passatgers, fet que permet que tant els passatgers del grup 1 com els del grup 2 realitzin l'embarcament de forma simultània.

Cas 1:

	A	B	C		D	E	F
25				2			
26				2			
27				1			
28				1			
29				1			
30	1	1	2	1	2	1	1

Cas 2:

	A	B	C		D	E	F
25				2			
26	2	2		2			
27				1			
28				1			
29				1			
30	1	1		1		1	1

Fig. 8.2 Efecte de l'assignació de files a diferents grups de passatgers

3- Si la fila anterior (*prevRow*) és igual a 1, llavors voldrà dir que l'últim seient haurà estat assignat a la primera fila. Això implica que caldrà reiniciar l'assignació de seients a partir de la última fila de l'avió (*currRow* = 30) en cas que el passatger actual no viatgi amb el passatger anterior o no hi hagi seients lliures a *prevRow*.

Exemple

Amb l'objectiu de mostrar de forma gràfica la metodologia d'assignació de seients descrita en l'apartat anterior, la següent figura (8.3) mostra un exemple de l'assignació de seients resultant en un vol amb un total de 144 passatgers, és a dir, un factor d'ocupació del 80%. Els números de cada cel·la representen l'identificador de cada grup de passatgers així com l'ordre en què aquests embarquen l'aeronau.

	A	B	C		D	E	F
1	30	30	43		43	43	30
2	16	16	29		42	16	16
3	15	28					
4	14						
5	13	27	62		62	62	27
6	12	52					61
7	26	26	52		52	52	41
8	11	11	41		41	41	41
9	10						
10	25	25			40	40	25
11	24	51			60	51	51
12	39	39	60		60	60	50
13	9	9	23		23	9	9
14	38	38					59
15	8	8	49		49	49	37
16	7						
17	6	48	58		58	58	48
18							
19	36	36			65	57	57
20	5	5	36		36	22	5
21	22	22	22		22	22	22
22	4	4	65		35	35	21
23	20	34	65		65	65	56
24	19	47	56		56	47	47
25	33	33	47		47	47	47
26	32	46			64	55	55
27	3	3	45		3	3	3
28	2	18			18	18	18
29	31	44			63	53	44
30	1	1	31		31	17	17

PASSADÍS

Fig. 8.3 Exemple de l'assignació de seients resultant mitjançant l'estratègia *ABP_RTSA*

9. Simulació del procés d'embarcament

Amb l'objectiu d'estudiar i avaluar el rendiment de les estratègies d'embarcament proposades en estudis anteriors i comparar-les amb les estratègies dissenyades en aquest estudi resulta necessari obtenir informació sobre el comportament que aquestes tindrien en l'operativa real d'una companyia aèria. En aquest cas, donat que no és possible dur a terme experiments reals, l'ús de tècniques de simulació resulta extremadament útil ja que permetrà estudiar les dinàmiques d'interès pròpies del procés d'embarcament de passatgers en aeronaus.

La simulació es podria definir com una tècnica que permet emular el comportament d'un sistema davant d'un conjunt de situacions determinades pel context en el que se situa el sistema. D'això se'n deriva que, mitjançant la simulació, serà possible estudiar el rendiment de diverses estratègies d'embarcament sota condicions inicials diferents mitjançant la definició de diversos escenaris, fet que permetrà extreure conclusions sobre el rendiment i eficiència de cada estratègia d'embarcament avaluada en l'estudi.

Donat que el procés d'embarcament de passatgers en aeronaus es pot considerar com un sistema dinàmic i estocàstic pel fet que el temps és una variable de gran importància i el comportament del sistema és desconegut a causa de la introducció de variables aleatòries, resulta útil utilitzar tècniques de simulació orientada a esdeveniments discrets. En aquest tipus de simulacions es modela el sistema o procés d'interès com una successió de diversos esdeveniments al llarg del temps i s'utilitzen variables d'estat, el valor de les quals canvia en instants no periòdics de temps, per tal de marcar l'evolució del procés al llarg del temps. Concretament, la simulació orientada a esdeveniments discrets es fonamenta en els següents elements:

- Esdeveniments: Cada esdeveniment representa una acció instantània que pot fer canviar l'estat del sistema.

- Activitats: Les activitats representen accions que tenen lloc en el sistema amb una duració temporal determinada. Cada acció es troba encapsulada entre dos esdeveniments.

- Llista d'esdeveniments: La llista d'esdeveniments conté tots els esdeveniments que encara no han tingut lloc a la simulació. Normalment, es tracta d'una llista dinàmica ordenada segons el temps previst d'execució de cada esdeveniment.

- Rellotge de la simulació: El rellotge permet representar el pas temporal entre la successió dels diferents esdeveniments. D'aquesta manera, el rellotge permet determinar l'ordre en què s'executaran cadascun dels esdeveniments de la llista.

Per tal de simular el procés d'embarcament, el simulador desenvolupat en aquest estudi parteix de la solució inicial obtinguda a partir de l'estratègia d'embarcament que es vulgui avaluar en cada cas. Concretament, la solució inicial conté la informació relativa al seient al qual ha estat assignat cada passatger, el grup d'embarcament al qual pertany i les parelles, famílies i grups que viatgen conjuntament. D'altra banda, abans d'iniciar la simulació es genera, de forma aleatòria, una cua de passatgers per tal de simular l'ordre en què aquests s'han disposat a la cua de les portes d'embarcament. És important senyalar que, a l'hora de generar la cua de passatgers, l'ordre dels grups d'embarcament es manté, de manera que només es distribueixen aleatòriament els passatgers dintre de cada grup, però en cap cas es barregen passatgers pertanyents a grups d'embarcament diferents. A més, al generar la cua de passatgers, també es tenen en compte aquells passatgers que viatgen de forma conjunta, de manera que mai són separats els uns dels altres.

Un cop generada la cua de passatgers, es crea un esdeveniment per a cada passatger amb l'objectiu d'emmagatzemar tota la informació relativa a l'estat d'aquest al llarg de tot el procés. Cada esdeveniment conté la següent informació:

- Temps: Indica l'instant de temps en el qual l'esdeveniment s'executarà dintre de la llista d'esdeveniments.

- Passatger: Conté tota la informació relativa al passatger al qual pertany l'esdeveniment. Entre d'altres, conté la informació relativa al número de fila i seient al qual ha estat assignat el passatger.

- Estat del passatger: Indica quin és l'estat actual del passatger. Concretament, el passatger pot trobar-se en un dels següents estats: caminant, desant l'equipatge de mà, resolent un conflicte de seient o ocupant el seient al qual ha estat assignat.

- Localització del passatger: Indica la posició actual del passatger dintre de la cabina de l'aeronau.

El conjunt d'esdeveniments de cada passatger forma la llista dinàmica d'esdeveniments del simulador, la qual s'ordena en funció del temps programat d'execució de cada esdeveniment. D'aquesta manera, el simulador executa l'esdeveniment que es troba al cap de la llista, n'actualitza el seu estat i l'elimina o insereix novament a la llista depenent de si el passatger ha ocupat, o no, el seient al qual ha estat assignat.

A continuació, en la figura 9.1 mostra la definició en JAVA de la classe "Event", la qual conté els atributs i mètodes de cada esdeveniment.

```
public class Event implements Comparable <Event> {

    private double time;

    private long eventCreationId = serial++;

    private Simulator.EventStatus status;

    private PAX pax;

    private int lastRow = 0;

    private int lastSeatPos = Simulator.AISLE;

    private static long serial = 0;


public Event(PAX pax) {

        this.pax = pax;

        this.time = 0;

        this.status = Simulator.EventStatus.WALKING;

    }

@Override

    public int compareTo(Event o) {

       int r = 0;

       if(time < o.time) r = -1;

       else if(time > o.time) r = 1;

       if(r == 0) {

             return (int) (eventCreationId - o.eventCreationId);

       }

       return r;

    }

}
```

Fig. 9.1 Definició en JAVA de cada esdeveniment del simulador

D'altra banda, per tal de controlar l'estat de la cabina de l'aeronau en cada instant de temps, resulta útil crear una matriu de dimensió 31x7 (anomenada `aisleEvents`) on s'hi representen el esdeveniments dels passatgers segons la seva localització en cada instant temporal. Això permet limitar el moviment dels passatgers que es troben bloquejats per altres passatgers i actualitzar el temps de l'esdeveniment del passatger

actual en funció del temps que trigarà un altre passatger en finalitzar l'activitat que està duent a terme.

D'aquesta manera s'aconsegueix representar les interferències causades entre els passatgers al llarg de tot el procés. La figura 9.2 mostra un fragment del simulador en què el passatgers de l'esdeveniment actual (`currentEvent`) es troba caminant pel passadís de l'avió.

```
if(currentEvent.getStatus() == EventStatus.WALKING) { //If the passenger of
the current event is walking:

  enqueue = true;

  if(currentEvent.getLastRow() != pax.getRow()) { //If the passenger hasn't
   reched his/her assigned row number:

      int nextRow = currentEvent.getLastRow()+1; //The next row is the current
         row+1

      Event eventInProgress = aisleEvents[nextRow][AISLE]; //The event in
       progress corresponds to the cell in front of the passenger's location

       if(eventInProgress == null) { //If there is not any event in the current
       event next position:

              aisleEvents[currentEvent.getLastRow()][AISLE] = null; //The
              current event position is released

              aisleEvents[nextRow][AISLE] = currentEvent; //The current event
              is set in the event in progress position

              currentEvent.setLastRow(nextRow); //The passenger's last row is
              updated

              currentEvent.incTime(pax.getSpeed()); //The event time is
              increased

              }

       else { //Else, the passenger has to wait until the eventInProgress is
       completed

              currentEvent.setTime(eventInProgress.getTime()+0.1);

              last = currentEvent;

              }

      }

  else { //If the passenger has reached his/her assigned row; he/she can
   proceed to store the hand luggage


       currentEvent.setStatus(EventStatus.STORING);

        currentEvent.incTime(pax.getStorage_Speed());

        }

  }
```

Fig. 9.2 Fragment del simulador en JAVA

Seguint el procés descrit anteriorment de forma iterativa i executant els esdeveniments de la llista fins que aquesta està buida és possible obtenir el temps total d'embarcament, el qual vindrà marcat pel temps d'execució de l'últim esdeveniment de la llista.

Finalment convé remarcar que, a part de la disposició aleatòria dels passatgers a la cua de les portes d'embarcament, el simulador té en compte altres elements estocàstics com són la velocitat de moviment de cada passatger, el temps necessari per a desar l'equipatge de mà i el temps necessari per a resoldre els conflictes de seient.

Secció 4: Implementació i validació de l'algorisme

10. Implementació de la solució en JAVA

Per tal d' implementar l'estratègia d'embarcament *ABP_SolGen* i comparar-la amb altres estratègies mitjançant l'ús de la simulació, resulta necessari desenvolupar un programa que permeti seguir els diferents passos definits en l'estratègia. Això permetrà obtenir una solució al problema d'embarcament de passatgers en aeronaus consistent en l'assignació de seients a cada passatger i la creació de grups d'embarcament d'acord amb els criteris definits en cada estratègia d'embarcament. Convé remarcar que l'ús de la programació, en aquest cas la programació informàtica mitjançant el llenguatge de programació orientat a objectes JAVA, no és més que un dels diversos instruments a través dels quals és possible implementar la seqüència de passos definits en l'apartat 7.

A continuació s'explica, de forma general, l'estructura i funcionament del programa utilitzat per a obtenir una solució al problema d'embarcament de passatgers en aeronaus d'acord amb una estratègia d'embarcament determinada. La seqüència de passos a través dels quals s'obté la solució és la següent:

1- Lectura de les dades d'entrada (inputs) del programa:

La lectura de dades d'entrada consisteix en processar tota la informació relativa al codi de reserva de cada passatger i l'any de naixement emmagatzemat en un fitxer .txt. Tota aquest procés es duu a terme en el mètode `read` (fig. 10.1) de la classe `InputReader` tal i com es mostra a continuació:

```
public LinkedList<PAX> read() {

        LinkedList<PAX> list = new LinkedList<PAX>(); //Linked list containing
        all passengers in the problem.

        try {

            BufferedReader br = new BufferedReader(new FileReader(fileName));

            Scanner in = new Scanner(br);



            int id = 1;

            in.nextLine(); //Needed in order to ignore the inputs file first
            row containing all data headers.



            /*Iterative creation of passengers (PAX) for each row introduced
             in the inputs file. The booking reference (ref)and year of birth
             (year) are read for each passenger*/



            while(in.hasNext()) {

                String ref = in.next();

                int year = in.nextInt();

                PAX p = PaxCreator.create(id++, year, ref); //Most of PAX
                class atributes are initialized in the PAX constructor
                depending on the year of birth (year) and the booking
                reference (ref).

                list.add(p);

            }

        } catch (IOException ex) {

            ex.printStackTrace();

        }

        return list;

    }
```

Fig. 10.1 Implementació en JAVA de la lectura de dades d'entrada

2- Creació dels passatgers d'acord amb la informació introduïda al fitxer d'inputs:

Per a cada línia introduïda en el fitxer de dades d'entrada cal crear un objecte de la classe `PAX` i definir el valor d'algun dels seus atributs en funció del codi de reserva i

any de naixement. Això es duu a terme mitjançant la classe `PAXCreator` (fig. 10.2), la qual es mostra a continuació:

```
public class PaxCreator {


   private static final double[][] distributionSpeedValues = { //Lower, Upper,
   Mode

         {0.8, 1.2, 1},

         {1,1.5,1.25}

  };

    private static final double[][] distributionStorageValues = { //Lower,
    Upper, Mode

         {5, 10, 7.5},

         {7.5,15,11.25}

  };

   public static PAX create(int id, int year, String ref) {

        GregorianCalendar cal = (GregorianCalendar) Calendar.getInstance();

        int currentYear = cal.get(GregorianCalendar.YEAR);

        int age = currentYear - year;

        int type = (age<=65)?1:2;

        double speedLower = distributionSpeedValues[type-1][0];

        double speedUpper = distributionSpeedValues[type-1][1];

        double speedMode = distributionSpeedValues[type-1][2];

        double speed = Utils.triangular(speedLower, speedUpper, speedMode);

        double storageLower = distributionStorageValues[type-1][0];

        double storageUpper = distributionStorageValues[type-1][1];

        double storageMode = distributionStorageValues[type-1][2];

        double    storage    =    Utils.triangular(storageLower,    storageUpper,
       storageMode);

        return new PAX(id, year,ref, type,age, speed, storage);

   }

}
```

Fig. 10.2 Creació de passatgers en JAVA

3- Creació de grups de passatgers en funció del codi de reserva per tal de representar als passatgers individuals, parelles, famílies i grups:

Amb l'objectiu de facilitar la manipulació dels passatgers que viatgen conjuntament resulta útil crear un objecte (`PAXgroup`) per a cada grup de passatgers amb codis de reserva iguals. Això es duu a terme al mètode `genInput` de la classe `InputReader`, tal i com es mostra en la figura 10.3:

```
public Input genInput(LinkedList<PAX> list) {

        HashMap<String, PAXGroup> dict = new HashMap<String, PAXGroup>();
        //HashMap relates a booking reference (String) with a set of
        passengers (PAXGroup) with this booking reference.

        //Allocation of passengers (p) into passenger groups (PAXGroup)
        according to the booking reference.

        for(PAX p: list) {

             if(!dict.containsKey(p.getBookingID())) //If there is not any
             passenger group with the selected passenger's booking reference,
             a new group of passengers is created.

                dict.put(p.getBookingID(), new PAXGroup(p.getBookingID()));

             PAXGroup group = dict.get(p.getBookingID()); //If there already
             exists a group of passengers with the passenger's (p) booking
             reference, the passenger is added to the group (group).

            group.add(p);

        }
```

Fig. 10.3 Creació de grups de passatgers

4- Classificació dels passatgers en funció de la dimensió del grup al qual pertanyen:

Un cop creats els grups de passatgers, resulta útil agrupar-los en funció de la seva dimensió per tal de facilitar l'assignació de seients i grups d'embarcament als passatgers. Per aconseguir-ho, s'assigna cada grup de passatgers a una llista en funció de la seva dimensió, tal i com es mostra en la figura 10.4.

```
LinkedList<PAXGroup> alones = new LinkedList<PAXGroup>(); //List of passengers
travelling alone

LinkedList<PAXGroup> pairs = new LinkedList<PAXGroup>(); //List of passenger
groups traveling in pairs.

LinkedList<PAXGroup> groups = new LinkedList<PAXGroup>();//List of passenger
groups of more than 2 members.

for(Entry<String, PAXGroup> e: dict.entrySet()) {

            switch(e.getValue().getSize()) { //Groups of passengers are added
            to one of the three lists depending on their size.

                case 1:

                    alones.add(e.getValue());

                    break;

                case 2:

                    pairs.add(e.getValue());

                    break;

                default:

                    groups.add(e.getValue());

                    break;

            }

        }
```

Fig. 10.4 Agrupament de grups de passatgers en funció de la seva dimensió

5- Assignació de seients i grups d'embarcament a passatgers en funció de l'estratègia d'embarcament i la dimensió de cada grup.

Per a cada passatger dintre de cada grup (`PAXGroup`), s'assigna un seient i un grup d'embarcament tenint en compte la política d'assignació establerta depenent de l'estratègia d'embarcament i el nombre de passatgers dintre de cada grup (un, dos, o tres o més).

6- Creació aleatòria de la cua d'embarcament respectant grups d'embarcament i passatgers que viatgen conjuntament.
7- Simulació del procés d'embarcament d'acord amb la política establerta en cada estratègia.

11. Verificació i validació de resultats

La verificació i validació del programa i del model de simulació resulten essencials per tal de poder aportar resultats fiables i de qualitat que permetin extreure conclusions ben fonamentades.

Per tal de garantir que la implementació del programa i del simulador són correctes s'han realitzat un conjunt de probes que han permès verificar el correcte funcionament del programa d'acord amb els criteris de construcció de la solució al problema d'embarcament de passatgers en aeronaus establerts per a cada estratègia d'embarcament avaluada. Concretament, s'han realitzat probes al llarg del desenvolupament del programa per tal de verificar el correcte funcionament de cadascun dels punts esmentats en l'apartat 10, fet que ha permès verificar la correcció de les solucions proposades en cada cas. D'altra banda, amb l'objectiu de verificar el correcte funcionament del simulador, a part de la meticulosa revisió del codi i dels diferents passos seguits en el procés, s'ha desenvolupat un visulitzador (veure fig. 11.1) per tal de representar de forma gràfica tot el procés d'embarcament. Això ha permès observar l'ordre en què els passatgers embarquen l'aeronau d'acord amb cada estratègia i verificar-ne la seva correcció. D'aquesta manera també ha estat possible verificar que, per exemple, tots els passatgers ocupen els seients als quals han estat assignats i l'ordre d'embarcament d'acord amb els grups d'embarcament establert es manté durant tot el procés. A continuació es mostra un exemple de la visualització:

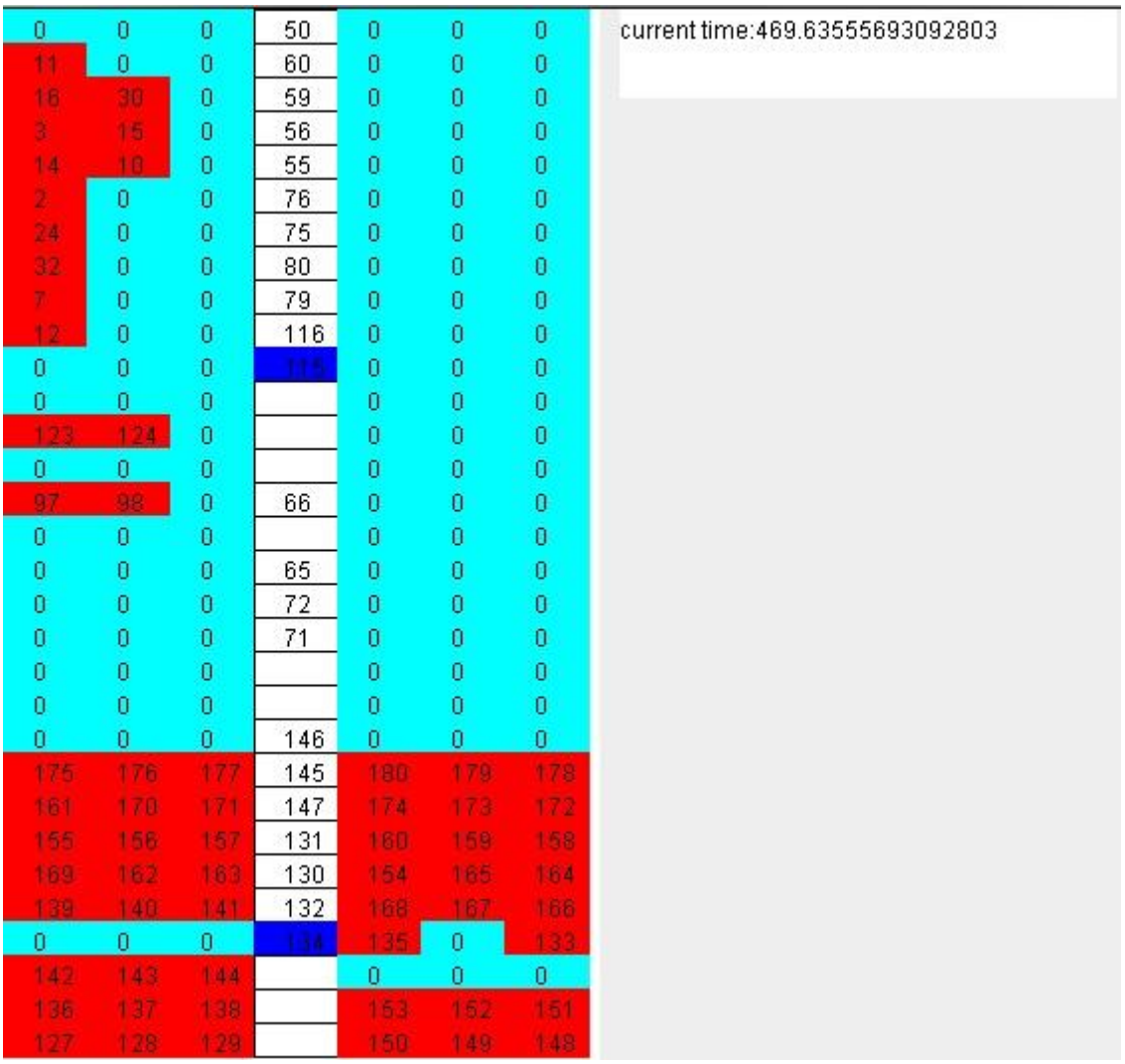


Fig. 11.1 Exemple del visualitzador del procés d'embarcament desenvolupat en JAVA

En relació a la validació del simulador, en primer lloc cal aclarir que aquest parteix d'un conjunt d'hipòtesis i supòsits inicials necessaris per a limitar la complexitat real del procés i del comportament propi de la naturalesa humana. Les principals hipòtesis sobre les quals s'ha desenvolupat el simulador són les següents:

- El nivell de puntualitat dels passatgers a les portes d'embarcament és del 100%. Això implica que tots els passatgers es troben a la cua quan aquests són cridats per embarcar l'aeronau.

- El nivell de compliment dels passatgers envers a l'estratègia d'embarcament utilitzada és del 100%, de manera que tots els passatgers ocupen els seients als quals han estat assignats i embarquen l'aeronau en el moment establert en funció del grup d'embarcament al qual pertanyen.

- Per a cada passatger, s'assumeix que la velocitat de desplaçament és constant durant tot el procés.

- Donat que l'estudi se centra en companyies aèries "Low-Cost", s'assumeix que tots els passatgers porten una peça d'equipatge de mà.

- No es produeixen avançaments entre els passatgers.

En segon lloc, amb l'objectiu d'obtenir temps d'embarcament propers a la realitat i reflectir les diferències existents entre cadascun dels passatgers, s'ha utilitzat una funció de distribució triangular per tal de determinar les velocitats de desplaçament i el temps necessari per a dipositar l'equipatge de mà en funció de la tipologia de cada passatger. Els paràmetres utilitzats en la funció triangular per a la velocitat dels passatgers s'han estimat d'acord amb els següents supòsits:

- La velocitat mitjana d'un individu jove oscil·la entre els 1,47 m/s i els 1,51 m/s.
- La velocitat mitjana d'un individu d'avançada edat oscil·la entre els 1,25 m/s i 1,31 m/s.
- La distància entre files en una aerolínea "Low-Cost" es, aproximadament, de 75 cm.
- Donades les limitacions d'espai a la cabina de l'aeronau, s'assumeix que els individus es desplacen més lentament que quan aquests caminen en condicions "normals".

Tenint en compte els supòsits esmentats anteriorment, els valors utilitzats en el simulador per a la velocitat de moviment dels passatgers (expressada en segons per fila) i el temps necessari per a dipositar l'equipatge de mà (expressat en segons) es mostren a continuació (fig. 11.2 i 11.3):

		Lower Bound	Mode	Upper Bound
		Type 1 PAX		
Triangular	Speed in aisle	0,8	1	1,2
		Type 2 PAX		
	Speed in aisle	1	1,25	1,5

Fig. 11.2 Taula dels valors utilitzats per a la velocitat de desplaçament dels passatgers

		Lower Bound	Mode	Upper Bound
		Type 1 PAX		
Triangular	Storage speed	5	7,5	10
		Type 2 PAX		
	Storage speed	7,5	11,25	15

Fig. 11.3 Taula dels valors utilitzats per al temps d'emmagatzematge de l'equipatge

Finalment, amb l'objectiu de validar les estimacions sobre el temps total d'embarcament obtingudes mitjançant el simulador, s'han realitzat diverses comparacions amb els resultats obtinguts en altres estudis i simuladors.

Secció 5: Experiments computacionals

12. Anàlisi de resultats

12.1 Introducció

Amb l'objectiu d'estudiar i extreure conclusions sobre el rendiment de l'estratègia d'embarcament *ABP_SolGen* i comparar-la amb altres estratègies, resulta oportú realitzar un conjunt de probes i anàlisis estadístics per tal de comparar els diversos resultats obtinguts a través de la simulació. A més, per tal de poder contrastar i avaluar el rendiment de cada estratègia d'embarcament sota condicions inicials diferents, s'han definit un conjunt d'escenaris que permetran observar si una estratègia d'embarcament concreta és més eficient que la resta independentment de les condicions inicials del problema o, per contra, algunes estratègies d'embarcament resulten ser més eficients que d'altres en determinades condicions o instàncies del problema.

Un cop obtingudes les observacions sobre els temps totals d'embarcament mitjançant la simulació en cada un dels escenaris caldrà comparar, per a cada instància del problema, els temps totals d'embarcament obtinguts mitjançant cada una de les estratègies d'embarcament per tal de concloure si existeixen, o no, diferències significatives en els temps totals d'embarcament depenent de l'estratègia emparada, fet que justificaria la necessitat d'utilitzar una estratègia d'embarcament concreta per tal de minimitzar el temps total d'embarcament. Donat que en l'estudi es compararan temps provinents de 4 estratègies d'embarcament diferents, l'ús d'un t-test no serà possible ja que s'haurà de comparar observacions provinents de més de dos grups, pel que serà necessari dur a terme un test ANOVA[5]. A més, com que es pretén comparar de quina manera afecta l'estratègia d'embarcament emprada (variable independent) en el temps total d'embarcament (variable dependent), el test ANOVA dependrà d'un únic factor ("One-way ANOVA"). Finalment, tal i com es detalla en el següent apartat, al haver-hi 8 instàncies del problema diferents caldrà realitzar un total de 8 tests ANOVA.

12.2 Disseny d'escenaris

Donat que l'estratègia d'embarcament *ABP_SolGen* ha estat especialment concebuda per a ser aplicable a l'operativa real i poder obtenir solucions eficients al problema d'embarcament de passatgers en aeronaus en funció de la tipologia dels passatgers d'un vol concret i les relacions existents entre aquests, s'han definit un conjunt

[5] ANOVA: El test ANOVA("Analysis of variance") s'utilitza per a concloure si existeixen diferències significatives entre les mitjanes obtingudes a partir de cada conjunt d'observacions provinents de tres o més grups. Això s'aconsegueix desglossant la variança total en la variança causada per factors aleatoris dintre de cada grup i la variança causada per diferències existents entre les observacions provinents de grups diferents.

d'escenaris diferents amb l'objectiu de conèixer i comparar el rendiment de les solucions proposades per cada estratègia d'embarcament en funció de diverses condicions inicials o instàncies diferents del problema. Més concretament, els factors que s'han tingut en compte per a obtenir els diferents escenaris han estat els següents:

- Factor d'ocupació: El factor d'ocupació és un valor percentual que indica la relació existent entre el nombre total de passatgers d'un vol en particular i la capacitat màxima de l'aeronau, el qual permet conèixer el nivell d'ocupació d'un vol determinat. Es tracta d'un factor important que cal tenir en compte per tal de conèixer el comportament de cada estratègia d'embarcament en funció del volum total de passatgers.
  Tenint en compte les conclusions extretes en l'estudi realitzat per Audenaert l'any 2009, en què s'observà que per sota d'un factor d'ocupació del 66% no existeixen diferències significatives en els temps totals d'embarcament obtinguts mitjançant estratègies diferents; en el disseny dels escenaris s'han considerat dos nivells de factors d'ocupació: 90% i 100%. Si bé és cert que el fet d'avaluar un factor d'ocupació del 100% pot resultar força poc realista, es considera interessant i necessari pel fet que permetrà establir una cota màxima del temps total d'embarcament assumint que la resta de factors es mantenen constants.

- Nivell de relació entre els passatgers: Donat que és evident que no tots els passatgers viatgen sols, resulta imprescindible tenir en compte el percentatge de passatgers que viatgen acompanyats i que, per tant, no poden ser separats al llarg de tot el procés d'embarcament. Es tracta d'un factor que ha estat ignorat en moltes de les probes realitzades en estudis anteriors però que resulta d'extremada importància ja que restringeix l'aplicabilitat d'algunes estratègies d'embarcament teòriques, com és el cas de l'estratègia *WILMA.* A més, en el disseny d'escenaris s'ha definit per a cada nivell diferent de relació entre els passatgers, el nombre total de passatgers que viatgen en parella i en grups de tres o més passatgers. Concretament, s'ha distingit entre dos nivells diferents de relació entre els passatgers: 70% i 85%. Es tracta de nivells de relació força elevats pel fet que l'estudi se centra en companyies aèries "Low-Cost", on s'assumeix que la majoria de les rutes són turístiques i de lleure, pel que el nombre de passatgers que viatgen de forma individual és baix.

- Tipologia dels passatgers: En l'estudi s'ha diferenciat entre dos tipus de passatgers en funció de la seva velocitat de moviment (tipus 1 o "Standard PAX" i tipus 2 o "Slow PAX"). Mentre que la velocitat del moviment de cada passatger pot estar condicionada per factors molt diversos, en una primera

aproximació s'ha pres l'edat com a criteri per a assignar un tipus a cada passatger. En particular, s'han definit dos nivells diferents de tipologia de passatgers mitjançant la variació del percentatge total de passatgers de cada tipus.

En la següent taula (fig. 12.1) es mostra un resum dels diferents valors definits per a cada un dels factors exposats anteriorment:

Factor d'ocupació	% PAX individuals	% PAX en parelles	% PAX en grups	% PAX tipus 1	% PAX tipus 2
100% (180 PAX)	30%	40%	30%	60%	40%
90% (162 PAX)	15%	40%	45%	80%	20%

Fig. 12.1 Taula dels valors utilitzats en el disseny d'escenaris

Tenint en compte que s'han definit dos nivells diferents per a cada un dels factors, els escenaris definits per a realitzar les diverses simulacions partiran de 8 instàncies diferents del problema (una per a cada combinació de factors possible). A més, per tal de poder comparar els resultats obtinguts mitjançant l'estratègia *ABP_SolGen*, se simularà cada instància del problema utilitzant cada una de les quatre estratègies d'embarcament esmentades a continuació:

- Estratègia *Random* i *Back-to-Front* per tractar-se de les estratègies d'embarcament amb més difusió i més utilitzades per les companyies aèries.

- Estratègia *Front-to-Back* perquè gran part dels estudis anteriors conclouen que es tracta d'una estratègia força poc eficient, fet que permetrà obtenir una cota màxima del temps total d'embarcament per a cada instància del problema.

- Estratègia *ABP_SolGen* per tractar-se de l'objecte principal de l'estudi i una de les aportacions originals d'aquest projecte.

El nombre total d'instàncies del problema junt amb cada una de les estratègies d'embarcament definides anteriorment resulta en un total de 32 escenaris diferents (8 instàncies del problema per a cada una de les 4 estratègies d'embarcament), el nom dels quals es mostra en la figura 12.2 d'acord amb la següent notació:

<Estratègia_%FactorOcupació_%NivellDeRelació_%Tipus1PAX>

Random_100_70_60	BackToFront_100_70_60	FrontToBack_100_70_60	ABPSolGen_100_70_60
Random_100_70_80	BackToFront_100_70_80	FrontToBack_100_70_80	ABPSolGen_100_70_80
Random_100_85_60	BackToFront_100_85_60	FrontToBack_100_85_60	ABPSolGen_100_85_60
Random_100_85_80	BackToFront_100_85_80	FrontToBack_100_85_80	ABPSolGen_100_85_80
Random_90_70_60	BackToFront_90_70_60	FrontToBack_90_70_60	ABPSolGen_90_70_60
Random_90_70_80	BackToFront_90_70_80	FrontToBack_90_70_80	ABPSolGen_90_70_80
Random_90_85_60	BackToFront_90_85_60	FrontToBack_90_85_60	ABPSolGen_90_85_60
Random_90_85_80	BackToFront_90_85_80	FrontToBack_90_85_80	ABPSolGen_90_85_80

12.2 Taula d'escenaris per a la simulació

D'altra banda convé recalcar que, per tal d'obtenir un volum significatiu d'observacions que permeti extreure conclusions a partir de la realització de probes estadístiques i l'anàlisi dels resultats obtinguts mitjançant la simulació, serà necessari executar 100 rèpliques de cada escenari, el que resultarà en un total de 3.200 simulacions.

Finalment, és important recordar que, a causa de la naturalesa estocàstica de la simulació originada per la introducció de factors aleatoris, la llavor inicial de la simulació per a cada escenari serà la mateixa. D'aquesta manera s'aconseguirà reduir al màxim les diferències entre observacions causades pels factors aleatoris (variança dintre de cada grup), fet que permetrà identificar amb més certesa les diferències entre observacions originades per diferències reals entre les diferents estratègies d'embarcament (variança entre grups).

12.3 Tests ANOVA

En el següent apartat es mostra, de forma detallada, les condicions inicials de cada una de les instàncies del problema així com els resultats obtinguts en els tests ANOVA mitjançant el software estadístic MINITAB. Donat que l'estructura és la mateixa per a cada instància del problema, s'ha considerat adient realitzar un comentari detallat en la primera instància i un comentari general sobre els resultats en el següent apartat.

Instància 90_70_60:

Condicions inicials del problema:

	#	PAX	%
Factor d'ocupació (%)		162	90%
Parelles	32	64	40%
Grups de 3 passatgers	6	18	11%
Grups de 4 passatgers	5	20	12%
Grups de 5 passsatgers	1	5	3%
Grups de 6 passatgers	1	6	4%
Grups de 7 passatgers	0	0	0%
Passatgers individuals	49	49	30%

Distribució dels passatgers segons la tipologia:

	#	Individual	Parelles	Grups
PAX tipus 1 (60%)	**97**	29	39	29
PAX tipus 2 (40%)	**65**	20	25	20

Distribució de la tipologia dels passatgers per a les parelles:

Tipologia dels integrants	# parelles	PAX tipus 1	PAX tipus 2
1_1	18	36	
1_2	3	3	3
2_2	11		22
Suma	**32**	**39**	**25**

Distribució de la tipologia dels passatgers per als grups:

Dimensió dels grups	# grups	PAX tipus 1	PAX tipus 2
3 PAX	6	9	9
4 PAX	5	20	
5 PAX	1		5
6 PAX	1		6
7 PAX	0		
Suma	**13**	**29**	**20**

Resultats del test ANOVA:

En primer lloc, és necessari verificar que es compleixen totes les hipòtesis necessàries per a poder dur a terme un test ANOVA. Aquestes són les següents:

1- Per a cada grup (per a cada estratègia d'embarcament), la variable dependent o resposta (temps total d'embarcament) segueix una distribució normal.

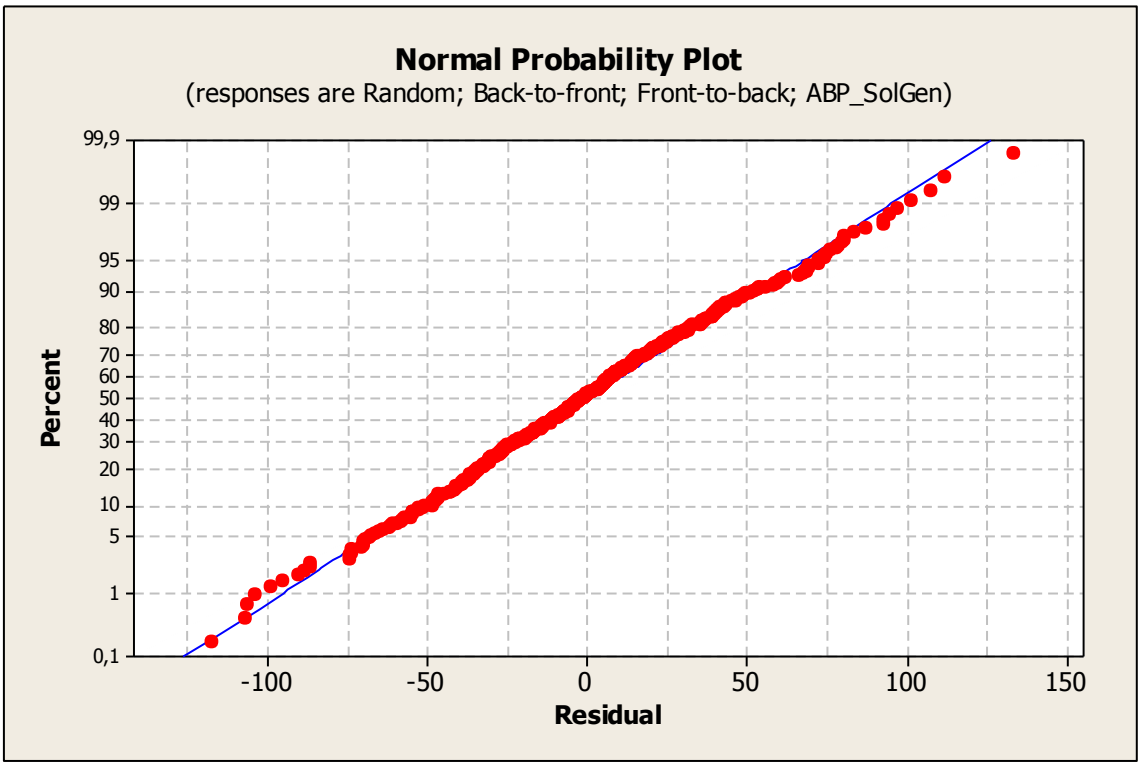


Fig. 12.3 Gràfic de normalitat de les observacions

En el gràfic de la figura 12.3 s'observa com tots els valors se situen a prop de la recta, fet que permet verificar la hipòtesi de normalitat de les variables.

2- La variança de la variable dependent és la mateixa per a tots els grups.

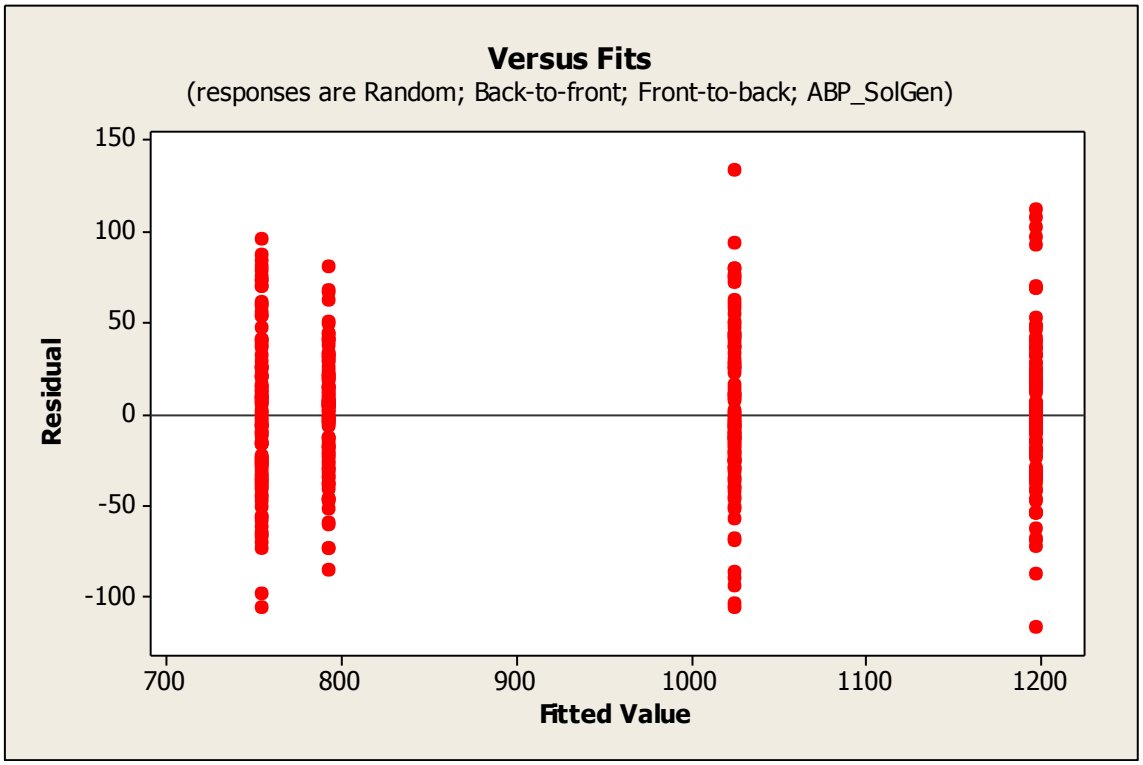


Fig. 12.4 Gràfic sobre la variança de les observacions per a cada grup

Per tal de verificar la hipòtesi de la variança, resulta extremadament útil plotejar els residus (la distància respecte al valor mig) de cada observació per a cada un dels grups. Tal i com s'aprecia en la figura 12.4, la longitud total de cada conjunt de residus corresponent a cada estratègia d'embarcament és aproximadament igual, fet que permet verificar la hipòtesi.

3- Les observacions són independents:

La introducció d'elements aleatoris per tal d'obtenir els valors dels temps totals d'embarcament permet verificar la hipòtesi d'independència de les observacions.

Un cop verificades les hipòtesis necessàries per a poder dur a terme un test ANOVA, resulta útil obtenir un gràfic de caixa (veure fig. 12.5) per a poder observar gràficament els temps mitjos obtinguts mitjançant cada estratègia d'embarcament. L'eix Y correspon al temps d'embarcament (en segons), mentre que l'eix X mostra cada estratègia d'embarcament. D'altra banda, la línia horitzontal corresponent a cada una de les caixes mostra el temps mig d'embarcament per a cada estratègia, mentre que els asteriscs representen la presència de "outliers", és a dir, observacions que s'allunyen molt dels valors mitjos.

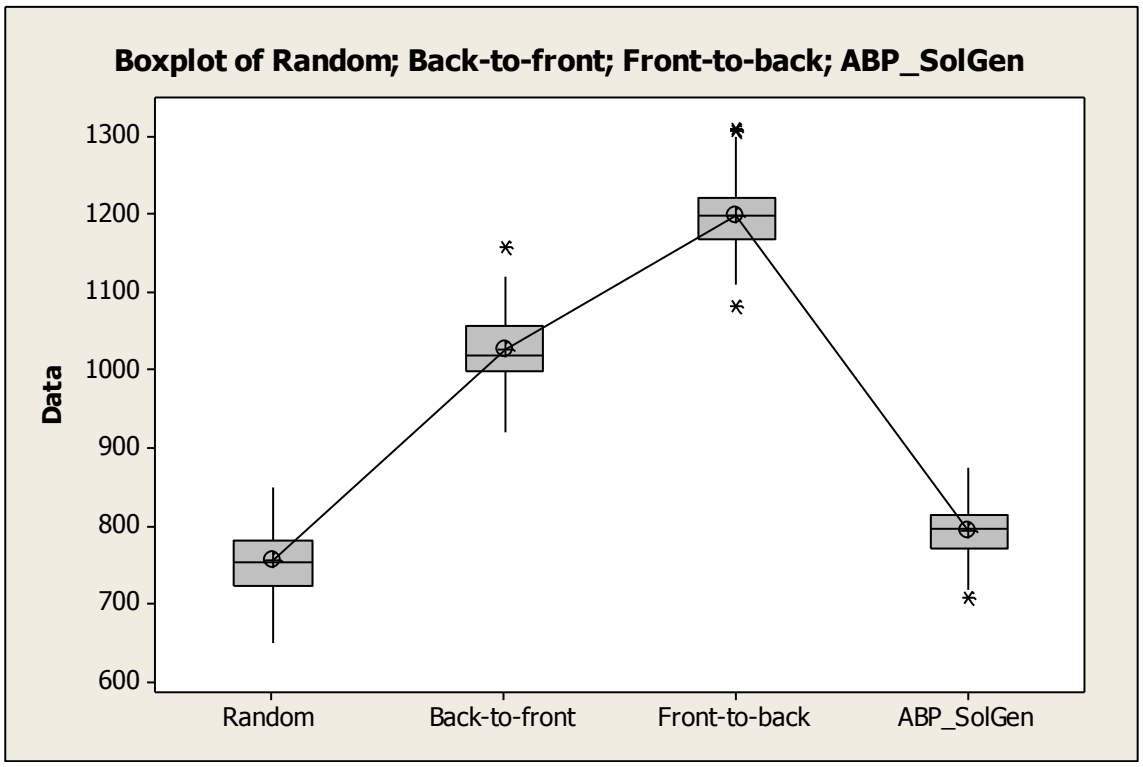


Fig. 12.5 Gràfic de caixa sobre els temps mitjos d'embarcament en l'escenari 90_70_60

En aquest cas s'observa que l'estratègia d'embarcament més eficient en termes de temps total d'embarcament és la *Random*, seguida per l'estratègia *ABP_SolGen, Back-to-front* i *Front-to-back*. A més, s'observa la presència de valors extrems en alguns dels temps totals d'embarcament obtinguts mitjançant les estratègies *Back-to-front*, *Front-to-back* i *ABP_SolGen*. És possible que aquest últim fet es degui al factor aleatori introduït per a crear l'ordre en què els passatgers accedeixen a l'aeronau en cada grup d'embarcament.

Tal i com es mostra a la figura 12.6, el P-valor obtingut en el test ANOVA és inferior a 0,05, pel que és possible rebutjar la hipòtesi nul·la que considera que les mitjanes de totes les estratègies d'embarcament són iguals. El fet de rebutjar la hipòtesi nul·la permet afirmar que almenys una de les mitjanes és diferent. No obstant, els "outputs" obtinguts amb MINITAB proporcionen la mitjana del temps total d'embarcament per a cada una de les estratègies d'embarcament, on s'observa que tots els valors són diferents, fet que permet afirmar que existeixen diferències significatives en els temps d'embarcament provinents de cada estratègia d'embarcament.

One-way ANOVA: Random; Back-to-front; Front-to-back; ABP_SolGen

```
Source    DF        SS        MS        F      P
Factor     3  12953109   4317703  2567,57  0,000
Error    396    665926      1682
Total    399  13619034

S = 41,01    R-Sq = 95,11%    R-Sq(adj) = 95,07%


                                    Individual 95% CIs For Mean Based on
                                    Pooled StDev
Level            N     Mean   StDev  --------+---------+---------+---------+-
Random         100    755,3    43,7  (*)
Back-to-front  100   1025,7    44,4                       *)
Front-to-back  100   1198,0    42,3                                        (*)
ABP_SolGen     100    793,2    32,5     (*)
                                    --------+---------+---------+---------+-
                                          840       960      1080      1200

Pooled StDev = 41,0
```

Fig. 12.6 Resultats del test ANOVA per a l'escenari 90_70_60

En la següent taula (fig. 12.7) es mostra el temps total d'embarcament mig necessari per a embarcar una aeronau sota les condicions inicials 90_70_60 així com el nombre mig d'interferències de seient originades en cada cas:

Estratègia	Temps d'embarcament	Conflictes de seient
Random	12m 35s	58,56
ABP_SolGen	13m 13s	17,53
Back-to-front	17m 5s	60,53
Front-to-back	19m 59s	59,97

Fig. 12.7 Taula resum dels resultats obtinguts en l'escenari 90_70_60

Instància 90_70_80:

Condicions inicials del problema:

	#	PAX	%
Factor d'ocupació (%)		162	90%
Parelles	32	64	40%
Grups de 3 passatgers	6	18	11%
Grups de 4 passatgers	5	20	12%
Grups de 5 passsatgers	1	5	3%
Grups de 6 passatgers	1	6	4%
Grups de 7 passatgers	0	0	0%
Passatgers individuals	49	49	30%

Distribució dels passatgers segons la tipologia:

	#	Individual	Parelles	Grups
PAX tipus 1 (80%)	**130**	39	52	39
PAX tipus 2 (20%)	**32**	10	12	10

Distribució de la tipologia dels passatgers per a les parelles:

Tipologia dels integrants	# parelles	PAX tipus 1	PAX tipus 2
1_1	26	52	
1_2	0	0	0
2_2	6		12
Suma	**32**	**52**	**12**

Distribució de la tipologia dels passatgers per als grups:

Dimensió dels grups	# grups	PAX tipus 1	PAX tipus 2
3 PAX	6	18	
4 PAX	5	16	4
5 PAX	1	5	
6 PAX	1		6
7 PAX	0		
Suma	**13**	**39**	**10**

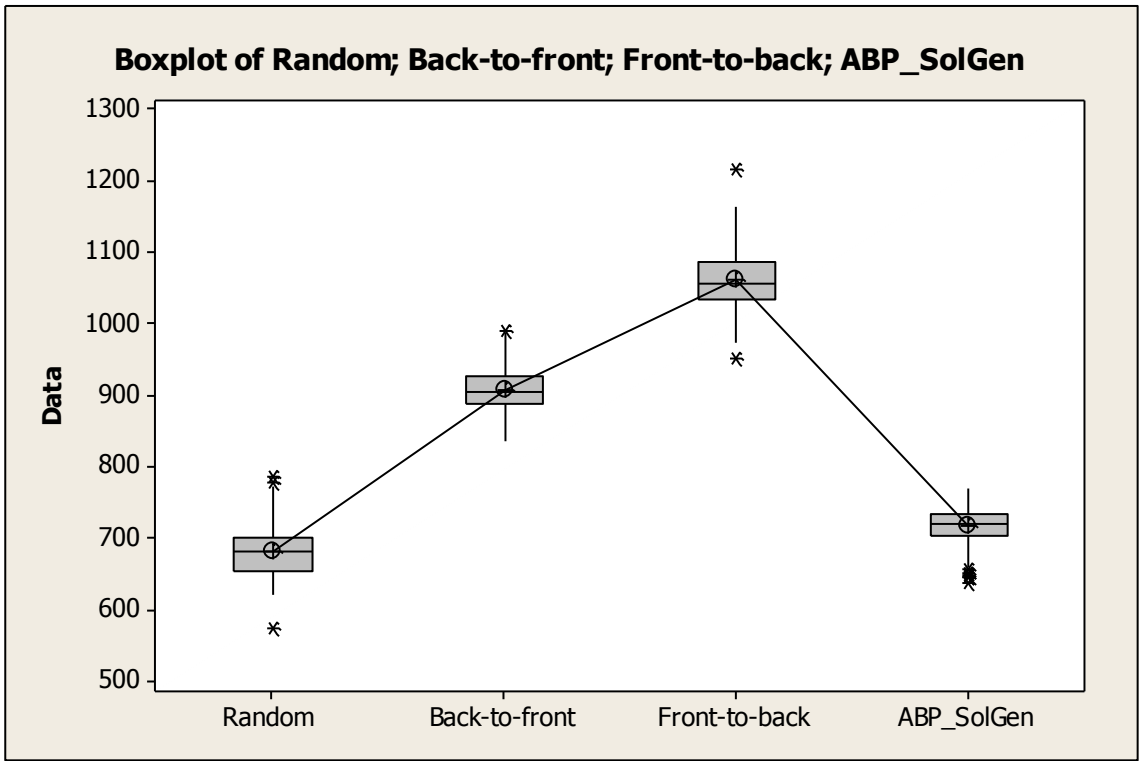


Fig. 12.8 Gràfic de caixa sobre els temps mitjos d'embarcament en l'escenari 90_70_80

One-way ANOVA: Random; Back-to-front; Front-to-back; ABP_SolGen

```
Source   DF       SS       MS        F      P
Factor    3  9327541  3109180  2426,51  0,000
Error   396   507411     1281
Total   399  9834951

S = 35,80   R-Sq = 94,84%   R-Sq(adj) = 94,80%


                                    Individual 95% CIs For Mean Based on
                                    Pooled StDev
Level            N    Mean  StDev  --+---------+---------+---------+-------
Random         100   683,5   40,2  *)
Back-to-front  100   908,8   33,1                    (*)
Front-to-back  100  1061,7   40,5                                      (*)
ABP_SolGen     100   717,8   27,8     (*
                                    --+---------+---------+---------+-------
                                    700       800       900      1000

Pooled StDev = 35,8
```

Fig. 12.9 Resultats del test ANOVA per a l'escenari 90_70_80

Estratègia	Temps d'embarcament	Conflictes de seient
Random	11m 23s	57,01
ABP_SolGen	11m 57s	17,53
Back-to-front	15m 8s	62,07
Front-to-back	17m 41s	60,29

Fig. 12.10 Taula resum dels resultats obtinguts en l'escenari 90_70_80

Instància 90_85_60:

Condicions inicials del problema:

	#	PAX	%
Factor d'ocupació (%)		162	90%
Parelles	32	64	40%
Grups de 3 passatgers	6	18	11%
Grups de 4 passatgers	5	20	12%
Grups de 5 passsatgers	2	10	6%
Grups de 6 passatgers	2	12	7%
Grups de 7 passatgers	2	14	9%
Passatgers individuals	24	24	15%

Distribució dels passatgers segons la tipologia:

	#	Individual	Parelles	Grups
PAX tipus 1 (80%)	**97**	15	38	44
PAX tipus 2 (20%)	**65**	9	26	30

Distribució de la tipologia dels passatgers per a les parelles:

Tipologia dels integrants	# parelles	PAX tipus 1	PAX tipus 2
1_1	17	34	
1_2	4	4	4
2_2	11		22
Suma	**32**	**38**	**26**

Distribució de la tipologia dels passatgers per als grups:

Dimensió dels grups	# grups	PAX tipus 1	PAX tipus 2
3 PAX	6	6	12
4 PAX	5	20	
5 PAX	2	5	5
6 PAX	2	6	6
7 PAX	2	7	7
Suma	**17**	**44**	**30**

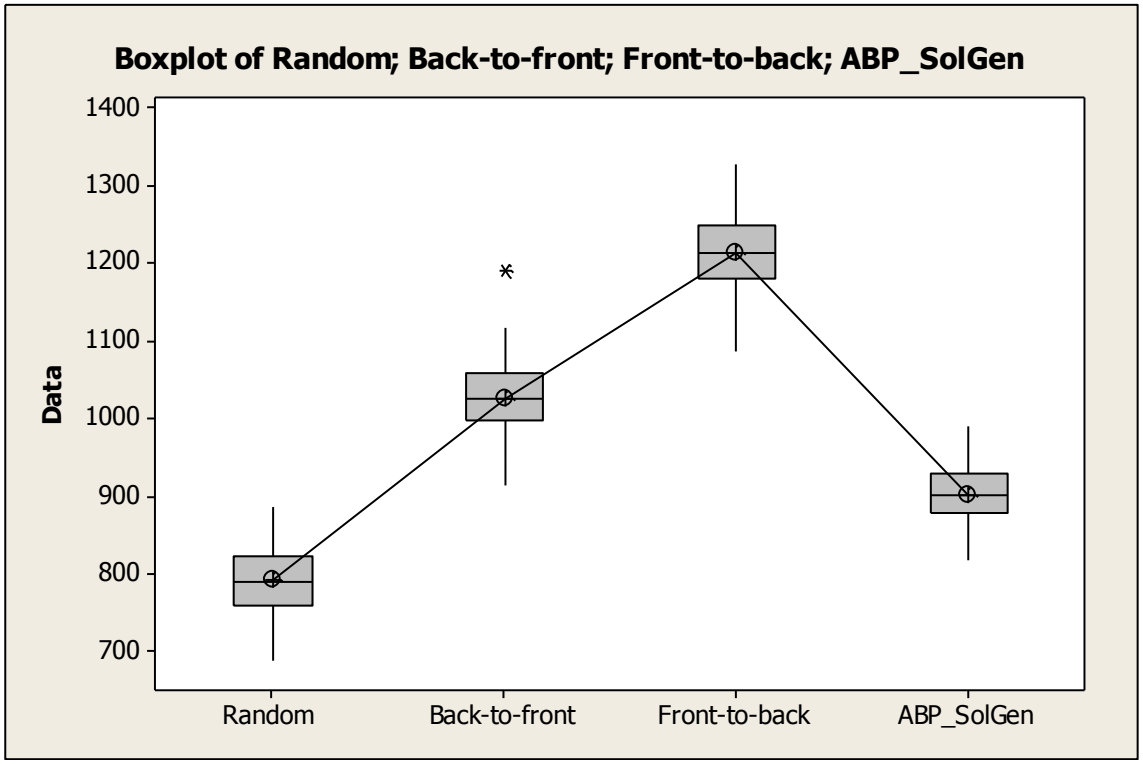


Fig. 12.11 Gràfic de caixa sobre els temps mitjos d'embarcament en l'escenari 90_85_60

One-way ANOVA: Random; Back-to-front; Front-to-back; ABP_SolGen

```
Source    DF        SS        MS         F      P
Factor     3   9868201   3289400   1642,31  0,000
Error    396    793152      2003
Total    399  10661353

S = 44,75    R-Sq = 92,56%    R-Sq(adj) = 92,50%


                                      Individual 95% CIs For Mean Based on
                                      Pooled StDev
Level             N     Mean   StDev  -----+---------+---------+---------+----
Random          100    792,4    46,8  (*)
Back-to-front   100   1027,4    44,6                    (*
Front-to-back   100   1215,3    49,6                                      *)
ABP_SolGen      100    902,7    37,1           (*)
                                      -----+---------+---------+---------+----
                                         840       960      1080      1200

Pooled StDev = 44,8
```

Fig. 12.12 Resultats del test ANOVA per a l'escenari 90_85_60

Estratègia	Temps d'embarcament	Conflictes de seient
Random	13m 12s	55,55
ABP_SolGen	15m 2s	26,77
Back-to-front	17m 7s	61,31
Front-to-back	20m 15s	60,76

Fig. 12.13 Taula resum dels resultats obtinguts en l'escenari 90_85_60

Instància 90_85_80:

Condicions inicials del problema:

	#	PAX	%
Factor d'ocupació (%)		162	90%
Parelles	32	64	40%
Grups de 3 passatgers	6	18	11%
Grups de 4 passatgers	5	20	12%
Grups de 5 passsatgers	2	10	6%
Grups de 6 passatgers	2	12	7%
Grups de 7 passatgers	2	14	9%
Passatgers individuals	24	24	15%

Distribució dels passatgers segons la tipologia:

	#	Individual	Parelles	Grups
PAX tipus 1 (80%)	**130**	20	52	58
PAX tipus 2 (20%)	**32**	4	12	16

Distribució de la tipologia dels passatgers per a les parelles:

Tipologia dels integrants	# parelles	PAX tipus 1	PAX tipus 2
1_1	24	48	
1_2	4	4	4
2_2	4		8
Suma	**32**	**52**	**12**

Distribució de la tipologia dels passatgers per als grups:

Dimensió dels grups	# grups	PAX tipus 1	PAX tipus 2
3 PAX	6	18	
4 PAX	5	16	4
5 PAX	2	5	5
6 PAX	2	12	
7 PAX	2	7	7
Suma	**17**	**58**	**16**

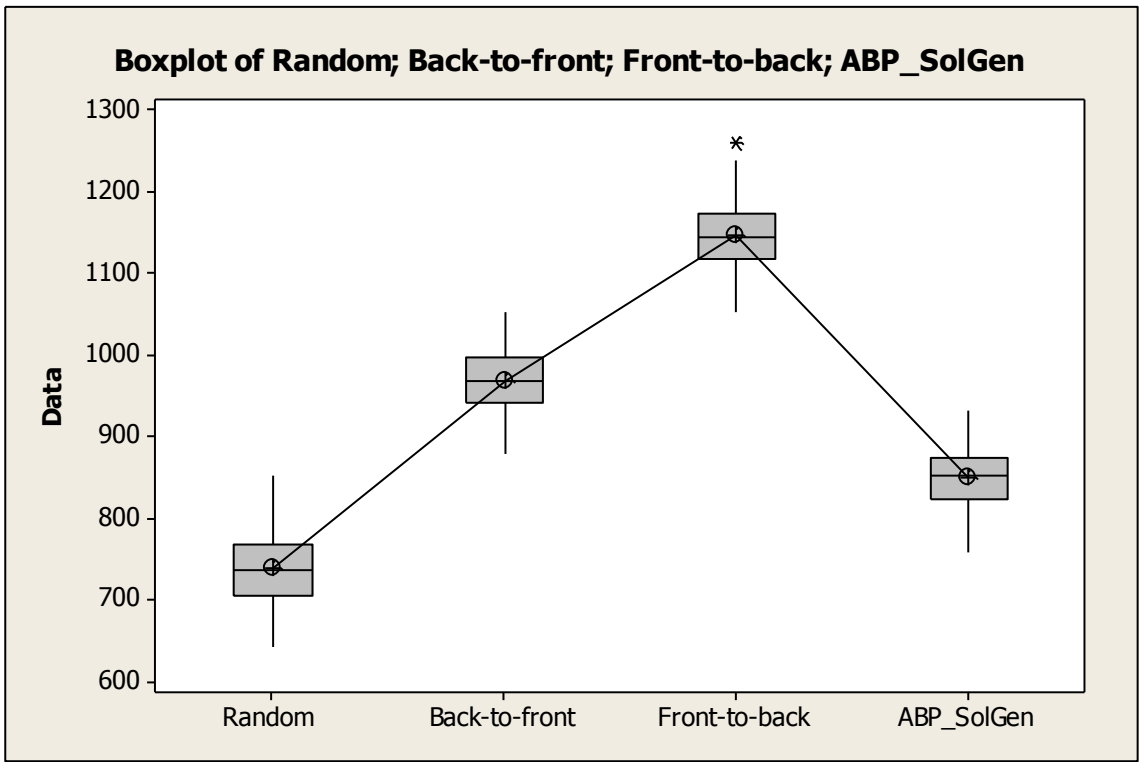


Fig. 12.14 Gràfic de caixa sobre els temps mitjos d'embarcament en l'escenari 90_85_80

One-way ANOVA: Random; Back-to-front; Front-to-back; ABP_SolGen

```
Source    DF       SS       MS         F      P
Factor     3  9044488  3014829  1828,63  0,000
Error    396   652878     1649
Total    399  9697365

S = 40,60   R-Sq = 93,27%   R-Sq(adj) = 93,22%
```

```
                                  Individual 95% CIs For Mean Based on
                                  Pooled StDev
Level            N    Mean  StDev  ---------+---------+---------+---------+
Random         100   740,2   47,5  (*
Back-to-front  100   968,7   38,8                  (*
Front-to-back  100  1146,2   39,9                                      (*
ABP_SolGen     100   851,1   35,3          (*)
                                  ---------+---------+---------+---------+
                                         840       960      1080      1200

Pooled StDev = 40,6
```

Fig. 12.15 Resultats del test ANOVA per a l'escenari 90_85_80

Estratègia	Temps d'embarcament	Conflictes de seient
Random	12m 20s	56,57
ABP_SolGen	14m 11s	26,77
Back-to-front	16m 8s	60,66
Front-to-back	19m 6s	59,45

Fig. 12.16 Taula resum dels resultats obtinguts en l'escenari 90_85_80

Instància 100_70_60:

Condicions inicials del problema:

	#	PAX	%
Factor d'ocupació (%)		180	100%
Parelles	36	72	40%
Grups de 3 passatgers	9	27	15%
Grups de 4 passatgers	4	16	9%
Grups de 5 passsatgers	1	5	3%
Grups de 6 passatgers	1	6	3%
Grups de 7 passatgers	0	0	0%
Passatgers individuals	54	54	30%

Distribució dels passatgers segons la tipologia:

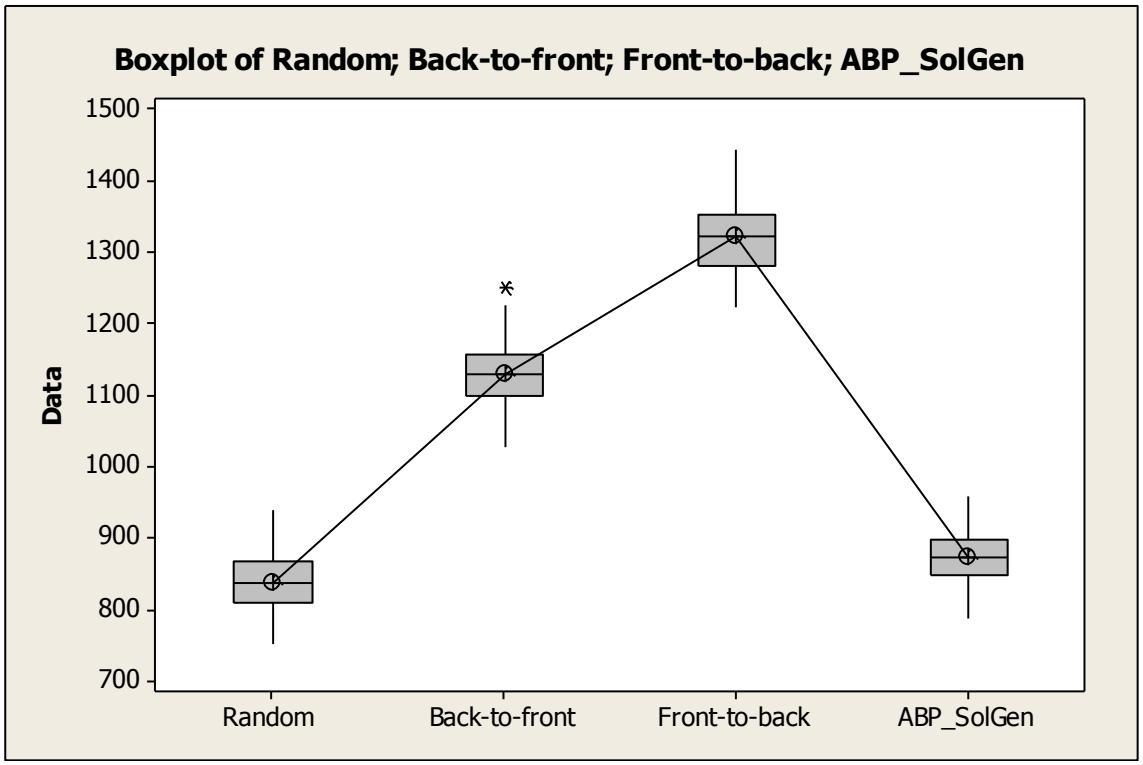
	#	Individual	Parelles	Grups
PAX tipus 1 (80%)	**108**	32	44	32
PAX tipus 2 (20%)	**72**	22	28	22

Distribució de la tipologia dels passatgers per a les parelles:

Tipologia dels integrants	# parelles	PAX tipus 1	PAX tipus 2
1_1	20	40	
1_2	4	4	4
2_2	12		24
Suma	**36**	**44**	**28**

Distribució de la tipologia dels passatgers per als grups:

Dimensió dels grups	# grups	PAX tipus 1	PAX tipus 2
3 PAX	9	21	6
4 PAX	4		16
5 PAX	1	5	
6 PAX	1	6	
7 PAX	0		
Suma	**15**	**32**	**22**



12.17 Gràfic de caixa sobre els temps mitjos d'embarcament en l'escenari 100_70_60

One-way ANOVA: Random; Back-to-front; Front-to-back; ABP_SolGen

Source	DF	SS	MS	F	P
Factor	3	15723050	5241017	2903,81	0,000
Error	396	714732	1805		
Total	399	16437782			

S = 42,48 R-Sq = 95,65% R-Sq(adj) = 95,62%

Level	N	Mean	StDev
Random	100	837,7	41,3
Back-to-front	100	1130,9	43,2
Front-to-back	100	1323,3	48,8
ABP_SolGen	100	873,3	35,5

Individual 95% CIs For Mean Based on Pooled StDev

```
(*
                    *)
                                  *)
   *)
-----+---------+---------+---------+----
   900      1050      1200      1350
```

Pooled StDev = 42,5

Fig. 12.18 Resultats del test ANOVA per a l'escenari 100_70_60

Estratègia	Temps d'embarcament	Conflictes de seient
Random	13m 57s	70,01
ABP_SolGen	14m 33s	19,47
Back-to-front	18m 50s	72,24
Front-to-back	22m 3s	70,54

Fig. 12.19 Taula resum dels resultats obtinguts en l'escenari 100_70_60

Instància 100_70_80:

Condicions inicials del problema:

	#	PAX	%
Factor d'ocupació (%)		180	100%
Parelles	36	72	40%
Grups de 3 passatgers	9	27	15%
Grups de 4 passatgers	4	16	9%
Grups de 5 passsatgers	1	5	3%
Grups de 6 passatgers	1	6	3%
Grups de 7 passatgers	0	0	0%
Passatgers individuals	54	54	30%

Distribució dels passatgers segons la tipologia:

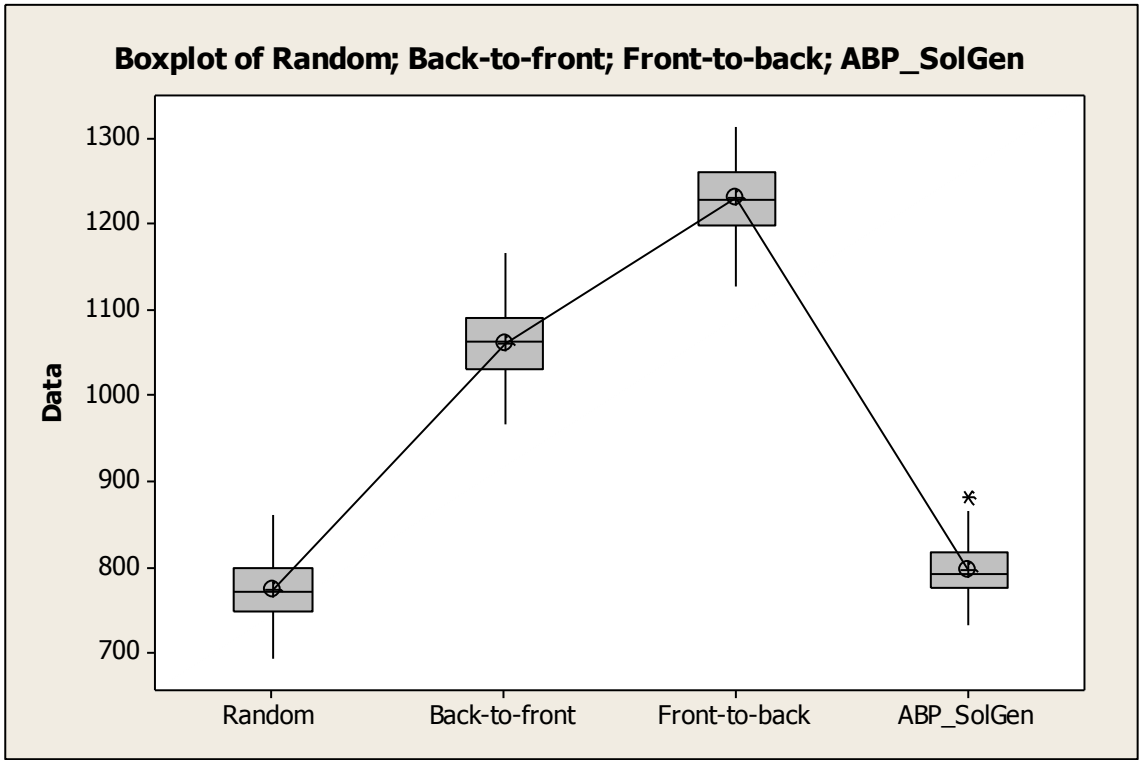
	#	Individual	Parelles	Grups
PAX tipus 1 (80%)	**144**	43	58	43
PAX tipus 2 (20%)	**36**	11	14	11

Distribució de la tipologia dels passatgers per a les parelles:

Tipologia dels integrants	# parelles	PAX tipus 1	PAX tipus 2
1_1	27	54	
1_2	4	4	4
2_2	5		10
Suma	**36**	**58**	**14**

Distribució de la tipologia dels passatgers per als grups:

Dimensió dels grups	# grups	PAX tipus 1	PAX tipus 2
3 PAX	9	24	3
4 PAX	4	8	8
5 PAX	1	5	
6 PAX	1	6	
7 PAX	0		
Suma	**15**	**43**	**11**



12.20 Gràfic de caixa sobre els temps mitjos d'embarcament en l'escenari 100_70_80

One-way ANOVA: Random; Back-to-front; Front-to-back; ABP_SolGen

```
Source    DF        SS        MS        F      P
Factor     3  14416988   4805663  3365,95  0,000
Error    396    565380      1428
Total    399  14982368

S = 37,79    R-Sq = 96,23%    R-Sq(adj) = 96,20%


                                    Individual 95% CIs For Mean Based on
                                    Pooled StDev
Level            N     Mean   StDev  ------+---------+---------+---------+---
Random         100    774,4    37,2  (*
Back-to-front  100   1060,6    41,1                        *)
Front-to-back  100   1230,4    41,7                                      (*
ABP_SolGen     100    796,8    29,8    *)
                                    ------+---------+---------+---------+---
                                        840       960      1080      1200

Pooled StDev = 37,8
```

Fig. 12.21 Resultats del test ANOVA per a l'escenari 100_70_80

Estratègia	Temps d'embarcament	Conflictes de seient
Random	12m 54s	69,25
ABP_SolGen	13m 16s	19,47
Back-to-front	17m 40s	72,59
Front-to-back	20m 30s	72,14

12.22 Taula resum dels resultats obtinguts en l'escenari 100_70_80

Instància 100_85_60:

Condicions inicials del problema:

	#	PAX	%
Factor d'ocupació (%)		180	100%
Parelles	36	72	40%
Grups de 3 passatgers	7	21	12%
Grups de 4 passatgers	6	24	13%
Grups de 5 passsatgers	2	10	6%
Grups de 6 passatgers	2	12	7%
Grups de 7 passatgers	2	14	8%
Passatgers individuals	27	27	15%

Distribució dels passatgers segons la tipologia:

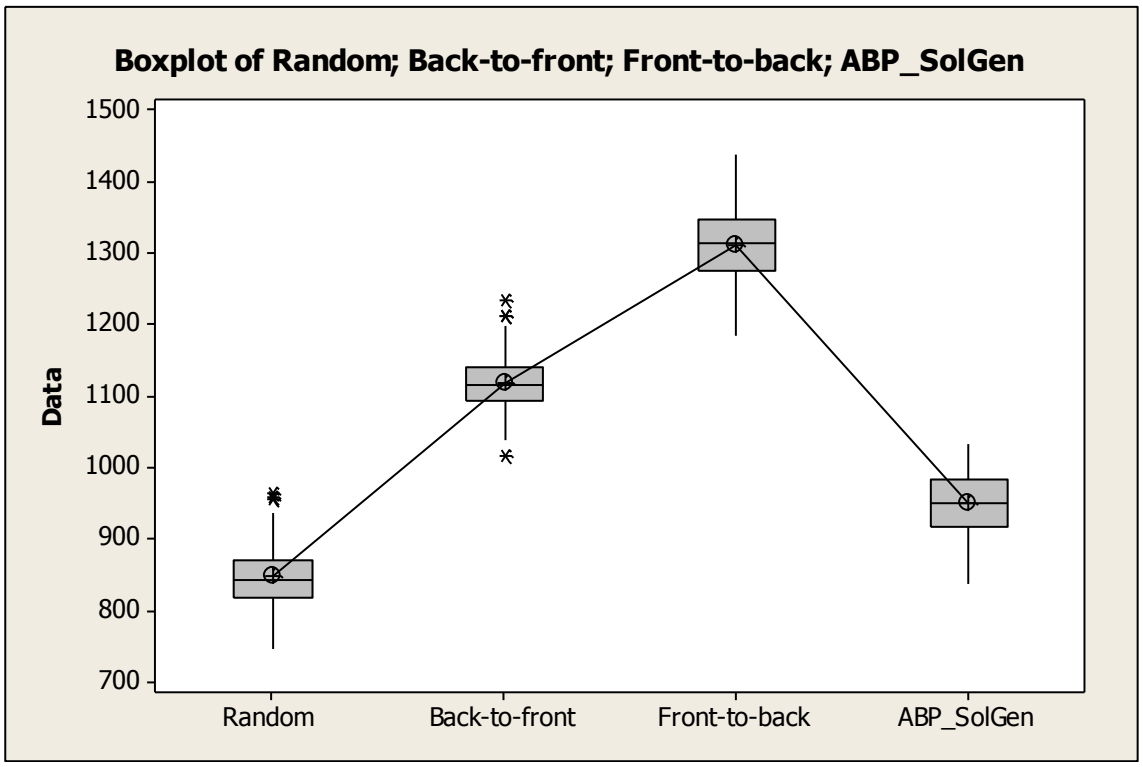
	#	Individual	Parelles	Grups
PAX tipus 1 (80%)	**108**	17	42	49
PAX tipus 2 (20%)	**72**	10	30	32

Distribució de la tipologia dels passatgers per a les parelles:

Tipologia dels integrants	# parelles	PAX tipus 1	PAX tipus 2
1_1	19	38	
1_2	4	4	4
2_2	13		26
Suma	**36**	**42**	**30**

Distribució de la tipologia dels passatgers per als grups:

Dimensió dels grups	# grups	PAX tipus 1	PAX tipus 2
3 PAX	7	18	3
4 PAX	6	12	12
5 PAX	2	5	5
6 PAX	2		12
7 PAX	2	14	
Suma	**19**	**49**	**32**



12.23 Gràfic de caixa sobre els temps mitjos d'embarcament en l'escenari 100_85_60

One-way ANOVA: Random; Back-to-front; Front-to-back; ABP_SolGen

```
Source    DF        SS       MS        F      P
Factor     3  12339604  4113201  2084,29  0,000
Error    396    781480     1973
Total    399  13121084

S = 44,42   R-Sq = 94,04%   R-Sq(adj) = 94,00%



Level            N    Mean  StDev
Random         100   849,8   42,5
Back-to-front  100  1119,1   40,2
Front-to-back  100  1312,1   51,2
ABP_SolGen     100   949,6   43,0

               Individual 95% CIs For Mean Based on Pooled StDev
Level            +---------+---------+---------+---------
Random           (*)
Back-to-front                          *)
Front-to-back                                          *)
ABP_SolGen               (*)
                 +---------+---------+---------+---------
               840       960      1080      1200

Pooled StDev = 44,4
```

Fig. 12.24 Resultats del test ANOVA per a l'escenari 100_85_60

Estratègia	Temps d'embarcament	Conflictes de seient
Random	14m 9s	70,29
ABP_SolGen	15m 49s	31,25
Back-to-front	18m 39s	70,59
Front-to-back	21m 52s	70,88

12.25 Taula resum dels resultats obtinguts en l'escenari 100_85_60

Instància 100_85_80:

Condicions inicials del problema:

	#	PAX	Share
Factor d'ocupació (%)		180	100%
Parelles	36	72	40%
Grups de 3 passatgers	7	21	12%
Grups de 4 passatgers	6	24	13%
Grups de 5 passsatgers	2	10	6%
Grups de 6 passatgers	2	12	7%
Grups de 7 passatgers	2	14	8%
Passatgers individuals	27	27	15%

Distribució dels passatgers segons la tipologia:

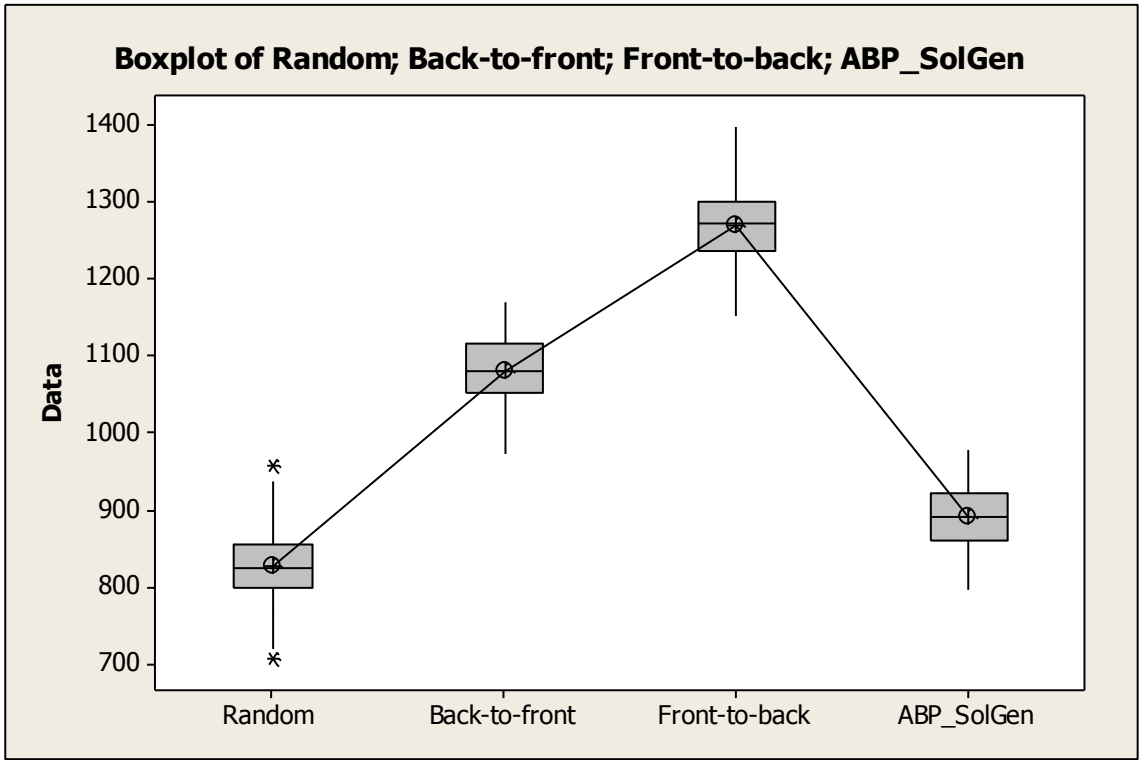
	#	Individual	Parelles	Grups
PAX tipus 1 (80%)	**144**	22	58	64
PAX tipus 2 (20%)	**36**	5	14	17

Distribució de la tipologia dels passatgers per a les parelles:

Tipologia dels integrants	# parelles	PAX tipus 1	PAX tipus 2
1_1	27	54	
1_2	4	4	4
2_2	5		10
Suma	**36**	**58**	**14**

Distribució de la tipologia dels passatgers per als grups:

Dimensió dels grups	# grups	PAX tipus 1	PAX tipus 2
3 PAX	7	18	3
4 PAX	6	20	4
5 PAX	2		10
6 PAX	2	12	
7 PAX	2	14	
Suma	**19**	**64**	**17**



12.26 Gràfic de caixa sobre els temps mitjos d'embarcament en l'escenari 100_85_80

One-way ANOVA: Random; Back-to-front; Front-to-back; ABP_SolGen

```
Source    DF        SS        MS        F      P
Factor     3  11941452   3980484  2068,38  0,000
Error    396    762079      1924
Total    399  12703532

S = 43,87   R-Sq = 94,00%   R-Sq(adj) = 93,96%


                                    Individual 95% CIs For Mean Based on
                                    Pooled StDev
Level            N    Mean   StDev  --+---------+---------+---------+-------
Random         100   826,6    46,0  (*)
Back-to-front  100  1081,4    42,6                    (*)
Front-to-back  100  1268,4    48,0                                      (*
ABP_SolGen     100   891,3    38,3        *)
                                    --+---------+---------+---------+-------
                                    840       960      1080      1200

Pooled StDev = 43,9
```

Fig. 12.27 Resultats del test ANOVA per a l'escenari 100_85_80

Estratègia	Temps d'embarcament	Conflictes de seient
Random	12m 35s	66,67
ABP_SolGen	14m 51s	31,25
Back-to-front	18m 1s	71,25
Front-to-back	21m 8s	70,9

12.28 Taula resum dels resultats obtinguts en l'escenari 100_85_80

12.4 Discussió dels resultats

Un cop obtingudes les estimacions sobre els temps totals d'embarcament per a cada escenari i després d'haver realitzat els tests ANOVA en cada una dels casos, és possible extreure una sèrie de conclusions sobre el comportament de les diferents estratègies d'embarcament.

En primer lloc, és possible observar que, en totes les instàncies del problema, el p-valor obtingut per al test global mitjançant el test ANOVA és inferior a 0,05, fet que permet rebutjar la hipòtesi nul·la que considera que les mitjanes de tots els grups són iguals. Això implica que almenys una de les quatre estratègies d'embarcament avaluades permet obtenir temps d'embarcament significativament diferents a la resta. No obstant, en tots els casos s'observa que els intervals de confiança sobre el temps totals d'embarcament per a cada una de les estratègies no intersecten entre sí , de manera que és possible afirmar que existeixen diferències significatives en els temps totals d'embarcament provinents de cada una de les estratègies. Aquest últim fet resulta d'extremada importància ja que permet justificar la necessitat d'utilitzar una estratègia determinada per tal de poder obtenir temps d'embarcament eficients. En definitiva, és possible afirmar que l'ús d'una estratègia o d'una altra no és trivial, sinó que realment és un element determinant a l'hora de reduir el temps total d'embarcament.

En segon lloc, una altra observació important és que l'ordre de les diferents estratègies d'embarcament en funció dels temps mitjos estimats es manté igual per a totes les instàncies del problema. Concretament, en tots els casos l'estratègia *Random* permet obtenir, en mitjana, els temps totals d'embarcament més baixos, seguida per l'estratègia *ABP_SolGen*, *Back-to-front* i *Front-to-back*. Aquest fet suggereix que, prenent com a únic element de judici el temps total d'embarcament, els diferents factors en funció dels quals s'ha dissenyat cada una de les instàncies del problema no afecten al rendiment de cada estratègia d'embarcament en relació a la resta. Això permet confirmar la robustesa de les estratègies d'embarcament eficients, donat que el seu rendiment no es veurà afectat per elements com el factor d'ocupació, el nivell de relació entre els passatgers ni la velocitat de moviment d'aquests. Per tant, és raonable considerar que, si una estratègia d'embarcament resulta ser més eficient que la resta en un cas concret, també ho serà en la resta de casos. En la figura 12.29 es mostra un gràfic amb els temps totals d'embarcament mitjos (en segons) obtinguts per a cada estratègia d'embarcament i instància del problema. Tal i com es pot observar, les línies corresponents al temps d'embarcament per a cada estratègia no es creuen en cap moment, fet que demostra l'explicació anterior.

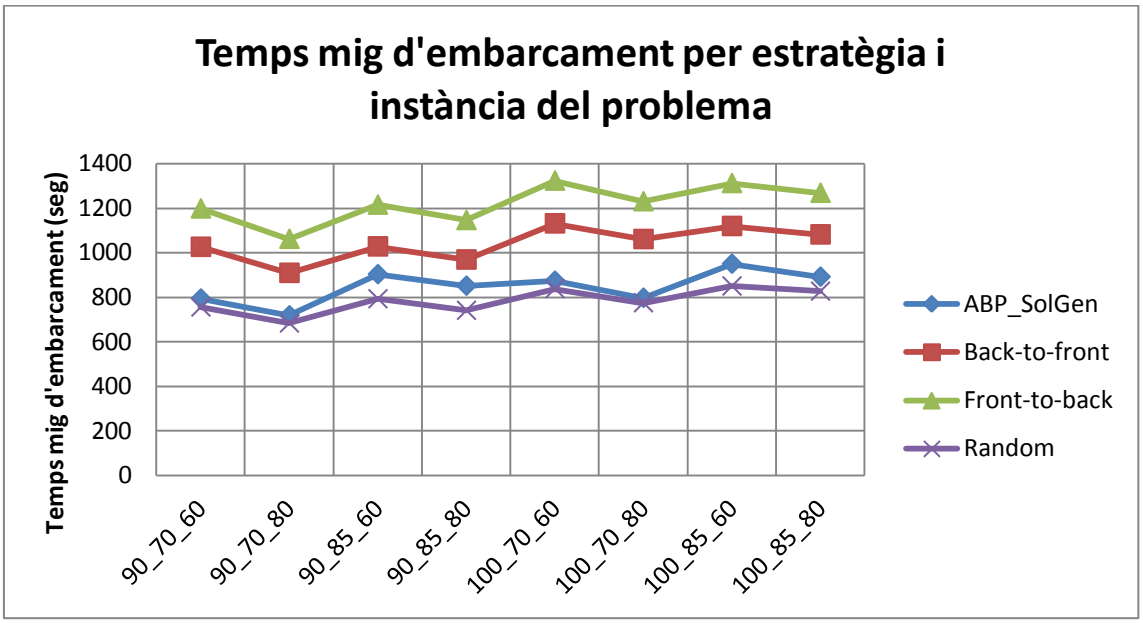


Fig. 12.29 Representació gràfica del temps mig d'embarcament per a cada estratègia i instància del problema

Si bé és cert que els diferents factors en funció dels quals s'han dissenyat les diferents instàncies del problema no afecten al rendiment de cada estratègia d'embarcament en relació a la resta, sí que tenen un impacte significatiu en el temps d'embarcament de cada estratègia. És per aquest motiu que, tal i com es pot observar en el gràfic, els temps d'embarcament més baixos tenen lloc en els casos en què el factor d'ocupació i el nivell de relació entre els passatgers és més baix i el nombre de passatgers del tipus 1 és més elevat. Concretament, els temps d'embarcament més baixos s'han obtingut en la instància 90_70_80.

El fet que l'estratègia *Random* sigui la que permet obtenir temps d'embarcament més eficients es deu, principalment, a dos factors derivats de l'ús d'un únic grup d'embarcament. D'una banda, el fet de no crear diferents grups d'embarcament permet distribuir de forma homogènia els passatgers dintre de la cabina de l'aeronau. Això té com a conseqüència un major aprofitament de l'espai disponible a la cabina de l'aeronau, fet que permet reduir el grau d'ociositat entre els passatgers i el nivell de densitat o concentració en zones determinades de l'avió. D'altra banda, la distribució homogènia dels passatgers al llarg de la cabina de l'aeronau permet maximitzar el nivell d'activitat simultània dels passatgers que estan ocupant els seients als quals han estat assignats. La combinació d'aquests dos factors resulta en una optimització de l'espai disponible a l'avió, fet que té com a conseqüència una reducció molt significativa del temps total d'embarcament.

Observant novament el gràfic de la figura 12.3, és possible apreciar com les diferències entre els temps totals d'embarcament obtinguts mitjançant les estratègies *Random* i

ABP_SolGen són considerablement petites. Particularment, les diferències entre les dues estratègies d'embarcament són especialment petites en els casos en què el nombre de passatgers individuals és més elevat, és a dir, en aquells casos en què el nivell de relació entre els passatgers és del 70%. Això es deu al fet que l'eficiència de l'estratègia d'embarcament *ABP_SolGen* és major a mesura que el nombre de passatgers individuals és més elevat ja que, en aquests casos, el nivell d'activitat simultània dintre de la cabina de l'aeronau serà major degut al criteri de creació dels grups d'embarcament.

Amb l'objectiu de conèixer més profundament el comportament de les diferents estratègies d'embarcament avaluades en l'estudi, resulta interessant estudiar els resultats obtinguts en relació al nombre total de conflictes de seient generats en cada escenari. En aquest cas resulta útil observar el gràfic de la figura 12.30, en què es mostra el nombre mig de conflictes de seient ocasionats en cada estratègia d'embarcament i per a cada instància del problema.

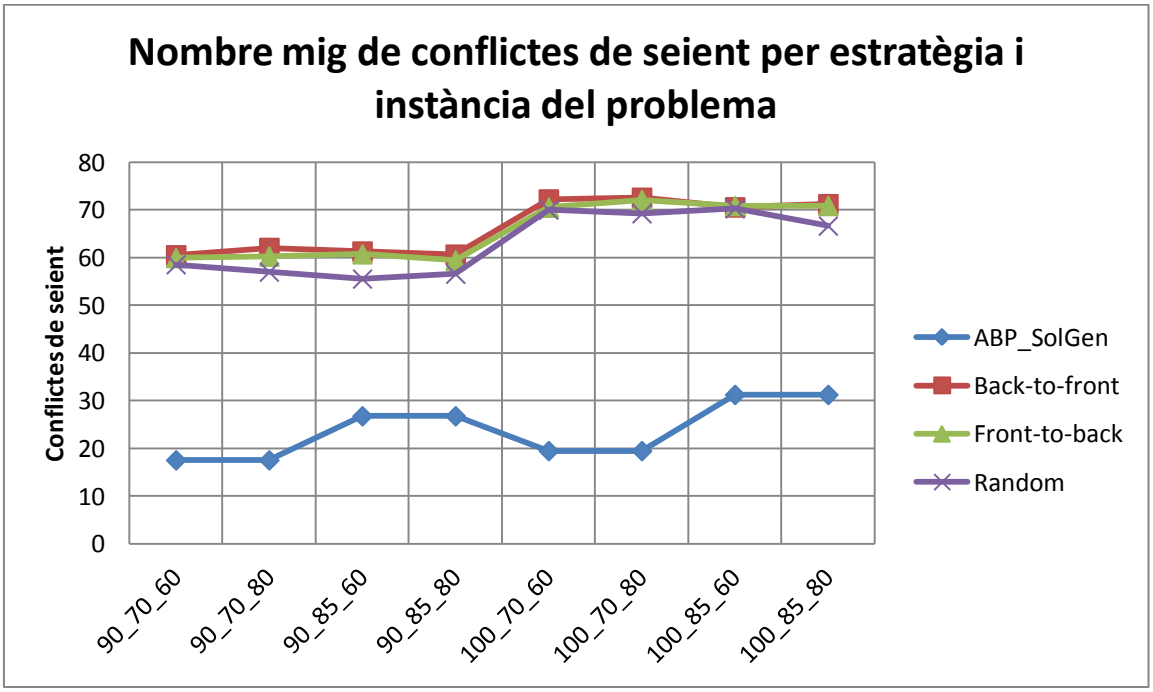


Fig. 12.30 Representació gràfica del nombre de conflictes de seient per a cada estratègia i instància del problema

Tal i com s'aprecia en el gràfic, el nombre de conflictes de seient ocasionats mitjançant l'estratègia *ABP_SolGen* és molt menor que els ocasionats en la resta d'estratègies. Això es deu al fet que es tracta d'una estratègia especialment dissenyada per a reduir al màxim aquest tipus de conflictes. D'altra banda, aquest fet permet afirmar que, de

totes les estratègies avaluades en l'estudi, l'estratègia *ABP_SolGen* és la que resultarà en una millor qualitat del servei percebut per part dels passatgers, fet que permetrà millorar el nivell de satisfacció d'aquests i augmentar el valor del servei ofert per part de la companyia aèria.

També resulta interessant observar el comportament del nombre total de conflictes de seient en funció de l'estratègia d'embarcament i la instància del problema. En el gràfic s'observa que en les estratègies *Back-to-front*, *Front-to-back* i *Random*, el factor d'ocupació afecta molt significativament al nombre total de conflictes de seient. D'alguna manera, resulta lògic assumir que, en aquests casos, el nombre total de conflictes de seient serà directament proporcional al nombre total de passatgers. No obstant, el comportament observat en el cas de l'estratègia *ABP_SolGen* és molt diferent. En aquest cas, el factor determinant que afecta de forma significativa al nombre de conflictes de seient és el nivell de relació entre els passatgers. Això es deu a que, en l'estratègia *ABP_SolGen*, només ocasionen conflictes de seient els passatgers que viatgen en grups de tres o més individus a causa del criteri utilitzat en la creació dels grups d'embarcament i l'ordre en què els passatgers embarquen l'aeronau. D'altra banda, en el gràfic s'observa que l'ordre de les diferents estratègies d'embarcament en funció del nombre total de conflictes de seient es manté igual per a totes les instàncies del problema. No obstant, a diferència dels resultats obtinguts en els temps totals d'embarcament, en aquest cas el nombre de conflictes de seient generats mitjançant les estratègies *Random*, *Back-to-front* i *Front-to-back* és molt similar.

Relacionant els resultats obtinguts sobre els temps totals d'embarcament i el nombre total de conflictes de seient ocasionats per a cada estratègia i instància del problema és possible extreure una sèrie de conclusions.

En primer lloc, prenent com a exemple els resultats obtinguts en l'estratègia *Random*, s'observa que, tot i que no és l'estratègia que ha ocasionat el menor nombre de conflictes de seient, es tracta de l'estratègia més eficient en quant al temps total d'embarcament. Això permet afirmar que un nombre elevat d'interferències (conflictes) entre els passatgers no implica necessàriament obtenir temps totals d'embarcament poc eficients. Tanmateix, és possible concloure que el impacte que tenen els conflictes ocasionats entre els passatgers sobre el temps total d'embarcament depèn, sobretot, de la localització d'aquests en la cabina de l'aeronau, és a dir, de com afecten a la resta dels passatgers.

En segon lloc, és possible observar que, mentre que l'estratègia *ABP_SolGen* és lleugerament més lenta que l'estratègia *Random*, les diferències existents en relació al nombre total de conflictes de seient ocasionats entre els passatgers són molt significatives. Això converteix l'estratègia d'embarcament *ABP_SolGen* en altament

eficient pel fet que permet obtenir temps d'embarcament molt semblants als de l'estratègia *Random* amb un nombre de conflictes entre els passatgers molt inferiors,

fet que té com a conseqüència una percepció del servei molt més positiva des del punt de vista de l'experiència dels passatgers. Aquest últim argument es veu reforçat per la política d'assignació de seients a passatgers dissenyada en l'estratègia *ABP_SolGen*, l'objectiu de la qual és assignar seients contigus al màxim nombre de passatgers que viatgen conjuntament. Tot això resulta en una situació "win-win" entre la companyia aèria i els passatgers ja que permet dur a terme el procés d'embarcament de forma ràpida sense malmetre la percepció de la qualitat del servei ofert als passatgers. És per aquest motiu que és possible concloure que l'estratègia *ABP_SolGen* suposaria una bona solució al problema d'embarcament de passatgers en aeronaus ja que permetrà dur a terme el procés d'embarcament de forma molt ràpida i sempre garantint un alt nivell de la qualitat del servei ofert als passatgers.

Secció 6: Conclusions i treball futur

13. Conclusions

L'embarcament de passatgers en aeronaus és un procés complex que depèn d'un nombre de factors molt elevat. Es tracta d'un problema especialment interessant ja que, tal i com s'ha esmentat al llarg de l'estudi, la seva optimització resulta en un benefici substancial per a les companyies aèries i els passatgers.

A partir de la revisió de la literatura científica s'ha pogut observar com el problema d'embarcament de passatgers en aeronaus ha estat tractat des de diverses perspectives i mitjançant metodologies tant diverses com la programació matemàtica, la simulació i l'ús d'algoritmes genètics, entre d'altres. També ha permès aprofundir en el problema i conèixer amb detall les diferents conclusions i solucions proposades en cada cas, fet que ha contribuït positivament a l'hora de dissenyar una estratègia d'embarcament eficient i realista.

Després d'estudiar el problema i les diverses estratègies d'embarcament existents, s'ha observat que algunes de les estratègies proposades en altres estudis són altament teòriques, ja que no tenen en compte les característiques individuals dels passatgers ni les relacions existents entre aquests, fet que en limita la seva aplicabilitat i validesa en l'operativa real. Això ha permès proposar dues estratègies d'embarcament (*ABP_SolGen* i *ABP_RTSA*) especialment dissenyades per a reduir al màxim el temps total d'embarcament i garantint-ne la seva aplicabilitat en l'operativa real tot assegurant un bon nivell de qualitat del servei ofert als passatgers.

A continuació s'ha desenvolupat un programa i un simulador mitjançant el llenguatge de programació JAVA amb l'objectiu d'assignar seients a passatgers, crear grups d'embarcament i obtenir estimacions realistes del temps total d'embarcament d'acord amb les estratègies *Random*, *Back-to-front*, *Front-to-back* i *ABP_SolGen* en funció de les condicions inicials del problema. Això ha permès realitzar un estudi comparatiu entre diverses estratègies d'embarcament i avaluar-ne el seu comportament i eficiència partint de diferents condicions inicials del problema.

Els resultats obtinguts mitjançant els tests ANOVA han permès confirmar la importància d'utilitzar una estratègia d'embarcament concreta, ja que s'ha demostrat que existeixen diferències significatives en el temps total d'embarcament depenent de l'estratègia emprada. D'altra banda, les estimacions sobre el temps total d'embarcament obtingudes mitjançant el simulador han permès concloure que, tal i com s'ha observat en altres estudis, l'estratègia d'embarcament *Random* és la que permet obtenir temps totals d'embarcament més baixos.

Finalment, les estimacions obtingudes en relació al temps total d'embarcament i nombre total de conflictes de seient han permès confirmar la gran eficiència de

l'estratègia *ABP_SolGen,* proposada en aquest estudi. S'ha observat que, mentre que el temps total d'embarcament és lleugerament superior al temps obtingut mitjançant l'estratègia *Random*, el nombre total d'interferències entre els passatgers és molt menor, fet que resulta en un augment de la qualitat del servei percebut per part dels passatgers.

14. Línies de treball futur

En aquest projecte s'ha desenvolupat un model de simulació que ha permès representar, de forma simplificada i partint d'un conjunt de supòsits inicials, el procés d'embarcament de passatgers en aeronaus per tal d'obtenir estimacions sobre el temps total d'embarcament i el nombre total d'interferències entre els passatgers depenent de l'estratègia d'embarcament utilitzada. Una possible línia de treball futur podria consistir en perfeccionar i refinar el model de simulació per tal de representar el procés d'embarcament de forma més realista mitjançant la introducció d'elements que permetin representar el comportament estocàstic i imprevisible de la naturalesa humana. Concretament, alguns dels canvis que es podrien realitzar són els següents:

- Introduir un factor aleatori que permeti reassignar el grup d'embarcament d'alguns passatgers i eliminar-ne d'altres per tal de representar la demora d'alguns passatgers a les portes d'embarcament, l'incompliment dels passatgers envers a l'estratègia d'embarcament utilitzada i la presència de "no-shows", és a dir, passatgers que no es presenten a les portes d'embarcament.

- Assignar el valor de la velocitat de desplaçament dels passatgers en funció de factors diferents a l'edat. Un exemple podria consistir en tenir en compte passatgers que viatgen amb nadons o passatgers que han de viatjar en cadira de rodes.

- Variar la velocitat de desplaçament dels passatgers en funció del nivell de congestió a la cabina de l'aeronau.

- Definir el temps necessari per a dipositar l'equipatge de mà en funció de les dimensions, pes i espai disponible als compartiments superiors de la cabina.

- Definir una funció que permeti estimar, de forma realista, el temps necessari per a resoldre un conflicte de seient entre dos passatgers.

D'altra banda, en el projecte s'han proposat dues estratègies d'embarcament diferents: *ABP_SolGen* i *ABP_RTSA*. No obstant, no ha estat possible avaluar el rendiment de l'estratègia *ABP_RTSA*, de manera que una altra línia de treball futur podria consistir en implementar l'algorisme de l'estratègia i sotmetre'l a diverses probes de simulació per tal d'avaluar-ne el seu rendiment i comparar-lo amb el d'altres estratègies d'embarcament.

Finalment, es podrien definir nous escenaris de simulació per tal d'estudiar el comportament de les diverses estratègies d'embarcament en funció d'altres factors i condicions inicials del problema que no s'han tingut en compte en aquest estudi. Entre d'altres, seria interessant estudiar el comportament de les estratègies d'embarcament en funció dels següents factors:

- Avaluar diferents tipus i dimensions d'aeronaus.
- Variar el nombre de passatgers amb equipatge de mà.
- Definir noves tipologies de passatger.
- Variar el nombre de portes a través de les quals els passatgers accedeixen a la cabina de l'avió.

Tot això permetria obtenir resultats a partir dels quals seria possible desenvolupar un model de regressió múltiple lineal per tal d'observar quins factors afecten, de forma significativa, al rendiment de cada una de les estratègies d'embarcament avaluades.

Referències i enllaços

15. Referències

Audenaert, J., 2009. "Multi-Agent Based Simulation for Boarding". *CODeS Research Group. The 21st Belgian-Netherland Conference on Artificial Intelligence. BNAIC (Eindhoven)*

Bachmat, E., 2005. "Analysis of airplane boarding via space-time geometry and random matrix theory". *Journal of Physics A: Mathematical and General, Vol. 39,* pp. 453-459.

Bazargan, M., 2007. "A linear programming approach for aircraft boarding strategy". *European Journal of Operational Research, 2007, Vol. 183,* pp. 394-411.

Cimler, R., 2012. "Agent-based model for comparison of aircraft boarding methods". *Proceedings of 30th International Conference Mathematical Methods in Economics.*

Ferrari, P., 2005. "Robustness of efficient boarding in airplanes" Nagel, K., *Transportation Research Record*, *2005*, pp. 44-54.

Inman, J., 2007. "QuickBoard: the Airplane Boarding Problem". *The Electronic Journal of Mathematics and Technology, 2007, Vol.1 (2)*.

Marelli, S., 1998. "The role of computer simulation in reducing airplane turn time. *Boeing Aero Magazine. No. 1.*

Nyquist, D.C., 2008. "A study of the airline boarding problem" K. L. McFadden, ed. *Journal of Air Transport Management, 2008, Vol. 14(4), pp.197-204*, 14(4), pp. 197–204.

Soolaki, M., 2011. "A new linear programming approach and genetic algorithm for solving airline boarding problem". *Applied Mathematical Modelling 2011, Vol. 36,* pp. 3949-4542.

Steffen, J.H., 2008. "Optimal boarding method for airline passengers". *Journal of Air Transport Management, 2008, Vol. 14(3), pp.146-150*, 14(3), pp.146–150.

Steffen, J.H., 2011. "Experimental test of airplane boarding methods". *Journal of Air Transport Management, 2012, Vol. 18,* pp. 64-67.

Steiner, A., 2009. "Speeding up the airplane boarding process by using pre-boarding areas". *Conference paper STRC 2009. 9th Swiss Transport Research Conference.*

Tang, T.-Q., 2012. "An aircraft boarding model accounting for passengers' individual properties" Y.-H. Wu, H.-J. Huang, & L. Caccetta, eds.*Transportation Research Part C, 2012, Vol. 22, pp.1-16*, 22, pp.1–16.

Van den Briel, M.H.L., 2005. "America West airlines develops efficient boarding strategies". *Interfaces, 2005, Vol. 35,* pp. 191-201.

Van Landeghem, H., 2002. "Reducing passenger boarding time in airplanes: A simulation based approach" A. Beuselinck, ed. *European Journal of Operational Research, 2002, Vol.142(2), pp.294-308, 142(2),* pp.294–308.

Wang, K., 2009. "Reducing Boarding Time: Synthesis of Improved Genetic Algorithms". *Proceedings of the 2009 Fifth International Conference on Natural Computation*, *2009*, *Vol. 5,* pp.403–407.

Yuhuan Cui, Y.C., 2009. "The Research of Optimization Model Based on Airplane Seating Problem" J. Q. Jingguo Qu, A. Y. Aimin Yang, & B. L. Baofeng Li, eds. *Innovative Computing, Information and Control, Dec. 2009, pp.511-514*, pp.511–514.

16. Enllaços

[1] http://www.icao.int/sustainability/pages/eap_fp_forecast_longterm.aspx

[2] http://www.filtcgilfoggia.it/notiziari_filt/2008/19/All19UI3.pdf

[3].http://www.airbus.com/presscentre/pressroom/presentations-speeches/?eID=dam_frontend_push&docID=25776

[4] http://www.boeing.com/commercial/cmo/

Menkes van den Briel: http://menkes76.com/projects/boarding/boarding.htm

Seatguru: http://www.seatguru.com/articles/boarding_procedures.php

Simulació: http://en.wikipedia.org/wiki/Discrete_event_simulation

Velocitat de desplaçament: http://en.wikipedia.org/wiki/Walking

Annex

17. Annex

A continuació s'adjunta el codi de programació en JAVA que constitueix el programa desenvolupat per a generar una solució al problema d'embarcament de passatgers en aeronaus i simular el procés d'embarcament. Per a cada classe es mostren els atributs i mètodes que la constitueixen.

- **ABP_SolGen.java**

```
package abp_solgen;


import abp_solgen.api.BoardingStrategy;

import abp_solgen.abpstrategy.ABPStrategy;

import abp_solgen.backtofront.B2FStrategy;

import abp_solgen.fronttoback.F2BStrategy;

import abp_solgen.randomstrategy.RandomStrategy;

import abp_solgen.utils.Utils;

import static abp_solgen.utils.Utils.ListShuffle;

import static abp_solgen.utils.Utils.setSeed;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.util.Iterator;

import java.util.LinkedList;

import java.util.logging.Level;

import java.util.logging.Logger;

import testrunner.Test;

/**

    The ABP_SolGen class represents the algorithm's main class where different
class methods are called in order to

    sequentially implement the ABP_SolGen heuristic.

 */
```

```
public class ABP_SolGen {


    private static final boolean DEMO_MODE = false;

    private static String inputPath = "";


    public static String getInputPath() {

        return inputPath;

    }


    public static void setInputPath(String path) {

        inputPath = path;

    }


   private final Test test;

   private final LinkedList<Solution> sols;

   private long timeTaken = 0;

   private ABP_SolGen(Test test) {

        this.test = test;


        //An InputReader class object (iReader) is created in order to read
        and pre-process all the input data from a .txt file

        InputReader iReader = new
        InputReader(getInputPath()+File.separatorChar+test.getFileName()+".txt
        ");

        Utils.setSeed(test.getSeed());


        Input input = iReader.genInput(iReader.read());


        BoardingStrategy currentStrategy = null;


        //[1=Random, 2=B2F, 3=F2B, 4=ABP]

        switch(test.getStrategy()) {

            case 1:

                currentStrategy = RandomStrategy.getStrategy();

                break;
```

```
            case 2:
                currentStrategy = B2FStrategy.getStrategy();
                break;
            case 3:
                currentStrategy = F2BStrategy.getStrategy();
                break;
            default:
                currentStrategy = ABPStrategy.getStrategy();
                break;
        }
        currentStrategy.assignBoardingGroup(input);

        double accum = 0;

        int sims = test.getnIters();

        sols = new LinkedList<Solution>();

        System.out.println("Running simulation "+test.getFileName()+"
        "+test.getnIters()+" "+test.getSeed()+" "+test.getStrategy());
        long t = System.currentTimeMillis();
        for(int i = 0; i < sims; i++) {
             Simulator sim = new Simulator(input, currentStrategy, DEMO_MODE);
             sim.run();
             Solution sol = Solution.newSol(i,
sim.getResult(),sim.getSeatInterferences());
                sols.add(sol);
        }
        long res = System.currentTimeMillis() - t;
        timeTaken = res;

       System.out.printf("Simulation completed in %d sec %n", (res/1000));
    }
```

```
public static ABP_SolGen launch(Test test) {

    ABP_SolGen instance = new ABP_SolGen(test);


    return instance;

}


public void saveTestTo(String outputs) {

    final String outputName = outputs + File.separatorChar +
    test.getFileName() + "_" + test.getStrategy() + "_" + test.getnIters()
    + "_" + test.getSeed() + ".txt";

    BufferedWriter bw = null;

    try {

        bw = new BufferedWriter(new FileWriter(outputName));


        String costList = "";

        double accumCost = 0;

        double accumInterference = 0;

        for(Solution s: sols) {

            accumCost += s.getCost();

            accumInterference += s.getInterferences();

            costList += s.toString()+"\n";

        }

        double avgCost = accumCost / test.getnIters();

        double avgInterferences = accumInterference / test.getnIters();


        int mm = (int) (avgCost / 60);

        int ss = (int) (avgCost % 60);

        bw.write("ABP SolGen result: \n");

        bw.write("Current scenario:\t"+test.getFileName()+"\n");

        bw.write("========================================\n");

        bw.write("Average boarding time (in min):\t"+mm+"m "+ss+"s\n");

        bw.write("Average boarding time (in sec):\t "+avgCost+" \n");

        bw.write("Average seat interferences:\t"+avgInterferences+" \n");

        bw.write("========================================\n");

        bw.write("Strategy:\t"+test.getStrategy()+"\n");
```

```
            bw.write("Seed:\t"+test.getSeed()+"\n");

            bw.write("nIters:\t"+test.getnIters()+"\n");

            bw.write("Execution time:\t"+(timeTaken/1000)+"sec\n");

            bw.write("=====================================\n");

            bw.write("BOARDING TIME LIST:\n");

            bw.write(costList);


        } catch (IOException ex) {


            Logger.getLogger(ABP_SolGen.class.getName()).log(Level.SEVERE,
            null, ex);

        } finally {

            try {

                bw.close();

            } catch (IOException ex) {


                Logger.getLogger(ABP_SolGen.class.getName()).log(Level.SEVERE,
                null, ex);

            }

        }

    }


}
```

- **Event.java**

```
package abp_solgen;

//The Event class is used in discrete event simulation to track the
current status and information of each passenger
// in the simulation and build a scheduled event list.
public class Event implements Comparable <Event> {

    private double time;
    private long eventCreationId = serial++;
    private Simulator.EventStatus status;
    private PAX pax;
    private int lastRow = 0;
    private int lastSeatPos = Simulator.AISLE; //All passengers start in
    the aisle of the aircraft cabin
    private static long serial = 0;
```

```
        @Override
        public String toString() {
               return "Event [time=" + time + ", eventCreationId=" +
               eventCreationId + ", status=" + status + ", pax=" + pax +
               ", lastRow="+ lastRow + ", lastSeatPos=" + lastSeatPos +
               "]";
        }

        public Event(PAX pax) {
          this.pax = pax;
          this.time = 0;
          this.status = Simulator.EventStatus.WALKING;
    }

    public void setStatus(Simulator.EventStatus status) {
        this.status = status;
    }

    public Simulator.EventStatus getStatus() {
        return this.status;
    }

    /**
       incTime method is used to increase and update the current event
time.
       @param time is the amount of time to be added in the time event.
     */
    public void incTime(double time) {
        this.time += time;
    }

    public void setTime(double time) {
        this.time = time;
    }


    public double getTime() {
        return time;
    }

    public PAX getPax() {
        return pax;
    }

    /**
       compareTo method compares the time of two different events.
       @param o represent the event to be compared to
       @return returns the time difference
     */
    @Override
    public int compareTo(Event o) {
       int r = 0;
       if(time < o.time) r = -1;
       else if(time > o.time) r = 1;
       if(r == 0) {
       return r;
    }

    public int getLastRow() {
        return lastRow;
    }

    public void setLastRow(int lastRow) {
        this.lastRow = lastRow;
    }
```

```
    public int getLastSeatPos() {
        return lastSeatPos;
    }

    public void setLastSeatPos(int lastSeatPos) {
        this.lastSeatPos = lastSeatPos;
    }


}
```

- **FlyRepr.java**

```
package abp_solgen;

import java.awt.Color;
import java.awt.Graphics;
import java.util.LinkedList;

import javax.swing.JFrame;
import javax.swing.JPanel;

public class FlyRepr extends JPanel {
	private static FlyRepr instance;

	private JFrame parent;
	public static FlyRepr getInstance() {
		if(instance == null) {
			JFrame f = new JFrame();
		  f.setTitle("SIM");
		  f.setVisible(true);
		  f.setSize(640, 640);
		  FlyRepr fr = new FlyRepr(f);
		  f.add(fr);
		  instance = fr;
		}
		return instance;
	}
	public FlyRepr(JFrame p) {
		parent = p;
	}

	public void redraw() {
		super.revalidate();
		super.repaint();
	}
	public int[][] current = new int[30][7];

	public void present(LinkedList<Event> events) {
		for(Event e: events) {
			current[e.getLastRow()][e.getLastSeatPos()]=
			e.getPax().getPAX_ID();
			if(e.getLastRow() > 25) {
				System.out.println(e);
			}
		}
		try {
			Thread.sleep(1);
		} catch (InterruptedException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	}
```

```
        public void paintComponent(Graphics g) {
                int w = 550;
                int h = 300;
                int rsw = w / 30;
                int rsh = h / 7;

                g.setColor(Color.white);
                g.fillRect(0, 0, h, w);
                g.setColor(Color.black);
                for(int i = 0; i < 30; i++) {
                        for(int j = 0; j < 7; j++) {
                                g.drawRect(j*rsh, i*rsw, rsh, rsw);
                                g.drawString(String.valueOf(current[i][j]),
j*rsh+15, i*rsw+15);
                        }
                }
        }
}
```

- **Input.java**

```
package abp_solgen;

import java.util.LinkedList;
import java.util.TreeMap;

/**
    Input class contains three different lists containing groups of
passengers travelling together. There is a list for each group size.
 */
public class Input {

    private LinkedList<PAXGroup> alone; //List containing 1 member
passenger groups.
    private LinkedList<PAXGroup> pairs; //List containing 2 member
passenger groups.
    private LinkedList<PAXGroup> groups;//List containing >2 memeber
passenger groups.
    private TreeMap<Integer, LinkedList<PAXGroup>> boardingGroups;
    private LinkedList<PAX> lista;

    /**
        Input class constructor sets each PAXGroup argument list to each
atribute list.
        @param alone contains a list of groups of passengers travelling
alone.
        @param pairs contains a list of groups of passengers travelling
in pairs.
        @param groups contains a list of groups of passengers of more
than 2 members.
     */
    public Input(LinkedList<PAXGroup> alone, LinkedList<PAXGroup> pairs,
LinkedList<PAXGroup> groups) {
        this.alone = alone;
        this.pairs = pairs;
        this.groups = groups;
         this.boardingGroups=            new            TreeMap<Integer,
        LinkedList<PAXGroup>>();
    }
```

```
    /**
        The registerGroup method adds a group of passengers into a list
        of passenger groups (bGroup) in order to create the
        different boarding groups.
        @param group represents the group of passengers to be added in
        the list.
     */

    public void registerGroup(PAXGroup group) {
        if(!boardingGroups.containsKey(group.getBoardingGroup()))
            boardingGroups.put(group.getBoardingGroup(),             new
            LinkedList<PAXGroup>());
           LinkedList<PAXGroup>                bGroup                =
           boardingGroups.get(group.getBoardingGroup());
           bGroup.add(group);
    }



    public TreeMap<Integer, LinkedList<PAXGroup>> getBoardingGroups() {
        return boardingGroups;
    }

    public LinkedList<PAXGroup> getPairs() {
        return pairs;
    }

    public LinkedList<PAXGroup> getGroups() {
        return groups;
    }

    public LinkedList<PAXGroup> getAlone() {
        return alone;
    }

    public LinkedList<PAX> getList() {
        return lista;
    }

    public void setAloneList(LinkedList<PAXGroup> list) {
        alone = list;
    }

    public void setPairsList(LinkedList<PAXGroup> list) {
        pairs = list;
    }

    public void setGroupsList(LinkedList<PAXGroup> list) {
        groups = list;
    }

    public void setCompleteList(LinkedList<PAX> lista) {
        this.lista = lista;
    }


    public void Ejemplo() {
        for(PAXGroup grupo: getGroups()) {
            for(PAX pasajero: grupo.getGroup()) {

            }
        }

    }
}
```

- **InputReader.java**

```
package abp_solgen;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.HashMap;
import java.util.LinkedList;
import java.util.Map.Entry;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author Gerard
 */

/**
    InputReader class reads all input data introduced in a .txt file and
creates all required objects to start and run the
    ABP_SolGen heuristic.
 */

public class InputReader {

    private final String fileName;

    /**
        InputReader class constructor.
        @param fileName contains the input file path.
     */
    public InputReader(String fileName) {
        this.fileName = fileName;
    }

    /**
        Creation of a PAX class object list from the data introduced in
the inputs file.
        @return: Returns a linked list (list) of class PAX objects.
     */
    public LinkedList<PAX> read() {
        LinkedList<PAX> list = new LinkedList<PAX>(); //Linked list
containing all passengers in the problem.
        try {
            BufferedReader br = new BufferedReader(new
FileReader(fileName));
            Scanner in = new Scanner(br);

            int id = 1;
            in.nextLine(); //Needed in order to ignore the inputs file
first row containing all data headers.

            /*Iterative creation of passengers (PAX) for each row
introduced in the inputs file. The booking reference (ref)
              and year of birth (year) are read for each passenger*/

            while(in.hasNext()) {
                String ref = in.next();
                int year = in.nextInt();
```

```
PAX p = PaxCreator.create(id++, year, ref); //Most of PAX class
atributes are initialized in the PAX constructor depending on the year
of birth (year) and the booking reference (ref).
                list.add(p);
            }

        } catch (IOException ex) {
            ex.printStackTrace();
        }

        return list;
    }


    /**
        Creation of groups of passengers travelling together (according
        to the booking reference) and subsequent creation of a list of
        groups according to the members belonging to each group (1, 2 or
        more than 2).

        @param list contains all passengers in the problem's instance
        (list of class PAX objects).
                @return: Returns an input class object containing three
        different lists of passenger groups.
     */

    public Input genInput(LinkedList<PAX> list) {
        HashMap<String, PAXGroup> dict = new HashMap<String,
        PAXGroup>(); //HashMap relates a booking reference (String) with
        a set of passengers (PAXGroup) with this booking reference.

        //Allocation of passengers (p) into passenger groups (PAXGroup)
        according to the booking reference.

        for(PAX p: list) {
            if(!dict.containsKey(p.getBookingID())) //If there is not
            any passenger group with the selected passenger's booking
            reference, a new group of passengers is created.

            dict.put(p.getBookingID(), new PAXGroup(p.getBookingID()));
            PAXGroup group = dict.get(p.getBookingID()); //If there
            already exists a group of passengers with the passenger's
            (p) booking reference, the passenger is added to the group
            (group).

            group.add(p);

        }

        //Creation of a list containing different passenger groups
        depending on the size of each group (1, 2 and more than 2).
LinkedList<PAXGroup> alones = new LinkedList<PAXGroup>(); //List of
passengers travelling alone
LinkedList<PAXGroup> pairs = new LinkedList<PAXGroup>(); //List of
passenger groups traveling in pairs.
LinkedList<PAXGroup> groups = new LinkedList<PAXGroup>();//List of
passenger groups of more than 2 members.
        for(Entry<String, PAXGroup> e: dict.entrySet()) {
            switch(e.getValue().getSize()) { //Groups of passengers are
            added to one of the three lists depending on their size.
                case 1:
                    alones.add(e.getValue());
                    break;
                case 2:
                    pairs.add(e.getValue());
                    break;
```

```
                    default:
                        groups.add(e.getValue());
                        break;
                }
            }
            Input input = new Input(alones, pairs, groups);

            input.setCompleteList(list);

            return input;
        }
        public static void main(String[] args) {
            File f = new File(".");
           System.out.println(f.getAbsolutePath());
           InputReader ir = new InputReader("inputs/input1.txt");
            //System.out.println("hola");
           ir.read();
        }
    }
```

- **PAX.java**

```
package abp_solgen;


/**

 PAX class represents each passenger in the ABP instance. A passenger has a
set of atributes and methods representing its traits.

 */


public class PAX

{

        private int PAX_ID; // PAX unique identificator

        private int birth_date;

         private int age;

         private int pax_type; // Passenger type based in the age (1 ==
         Standard; 2 == Slow)

         private boolean luggage = true; // True == The passenger is carrying
         hand luggage

         private String booking_ID; // The booking reference

         private int pax_row; // Row number assigned to the passenger

         private int pax_seat; // Seat letter assigned to the passenger

         private double speed; // The passenger's movement speed while boarding

         private double storage_speed; // The passenger's speed while storing
         luggage
```

```
    private double sitting_time; // The time it takes the passenger to sit


public PAX (int i, int year, String booking, int type, int age,double
speed, double storage)

{

    PAX_ID = i;

    birth_date = year;

    this.age = age;

    this.pax_type = type;

    this.speed = speed;

    sitting_time = speed;

    booking_ID = booking;

    storage_speed = storage;

}


public void setAge(int birth_date)

{

    age = birth_date;

}


public void setPAX_type()

{

    if (age<65){


        pax_type = 1;

    }

    else

        pax_type = 2;

}


public void setSeat(int row, int seat) {

    this.pax_row = row;

    this.pax_seat = seat;

}
```

```
public String getBookingID() {
    return booking_ID;
}
public int getPAX_ID()
{
    return PAX_ID ;
}

  public int getType()
{
    return pax_type;
}

  public int getRow()
{
    return pax_row ;
}

  public int getSeat()
{
    return pax_seat ;
}


  public double getSpeed()
{
    return speed ;
}

  public double getStorage_Speed()
{
    return storage_speed ;
}
```

```
      public double getSitting_Time()

    {

        return sitting_time ;

    }


    @Override

    public String toString() {

        return "PAX{" + "PAX_ID=" + PAX_ID + ", birth_date=" + birth_date + ",
        age=" + age + ", pax_type=" + pax_type + ", luggage=" + luggage + ",
        booking_ID=" + booking_ID + ", pax_row=" + pax_row + ", pax_seat=" +
        pax_seat + ", speed=" + speed + ", storage_speed=" + storage_speed +
        ", sitting_time=" + sitting_time + '}';

    }

}
```

- **PAXGroup.java**

```
package abp_solgen;


import java.util.Comparator;

import java.util.LinkedList;


/**

    PAXGroup class represents a set of passengers traveling together. A group
of passengers is set according to the booking reference.

    Hence, passengers with the same booking reference will belong to the same
group.

 */

public class PAXGroup implements Comparable<PAXGroup> {


    private String bookRef;

    private LinkedList<PAX> list;

    private int boardingGroup;
```

```
/**

    The PAXGroup constructor sets the booking reference identifying each
    group and creates a new list of passengers.

    @param bookRef is the booking reference of all passengers belonging to
    the same group.

 */

public PAXGroup(String bookRef) {

    this.bookRef = bookRef;

    this.list = new LinkedList<PAX>();

}

/**

    The add method adds a PAX object (p) to the list of passengers.

    @param p is a class PAX object.

 */

public void add(PAX p) {

    list.add(p);

}

/**

 * The compareTo method compares the size of two different passenger
   groups

 * @param o represents a group of passengers

 * @return returns the difference in the size of the passenger groups.

 */

@Override

public int compareTo(PAXGroup o) {

    return getSize() - o.getSize();

}

/**

    @return: Returns the group size.

 */

public int getSize() {

    return list.size();

}
```

```
/**
    @return: Returns a group of passengers.
 */
public LinkedList<PAX> getGroup() {
    return list;
}
/**
    @return: Returns the group's booking reference.
 */
public String getBookRef() {
    return bookRef;
}
/**
    Sums the type of all passenger's belonging to the same group in order
    to enable a type-based passenger seat allocation.
    @return: Returns the type sum of all passengers in the group.
 */
public int getTypeSum() {
    int r = 0;
    for(PAX p : list)
        r += p.getType();

    return r;
}

public PAX getFirst() {
    return list.getFirst();
}

public PAX getLast() {
    return list.getLast();
}
public int getBoardingGroup() {
    return boardingGroup;
}
```

```
    public void setBoardingGroup(int group) {

        this.boardingGroup = group;

    }

    /**

       Defines a type-based sorting criteria in order to compair two different
       PAXGroup objects.

     */

    public    static    final    Comparator<PAXGroup>    TYPE_SORT    =    new
    Comparator<PAXGroup>() {

        @Override

        public int compare(PAXGroup o1, PAXGroup o2) {

           return o1.getTypeSum() - o2.getTypeSum();

        }

    };

}
```

- **PAXCreator.java**

```
package abp_solgen;

import abp_solgen.utils.Utils;
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;

/**
 *
 * @author Gerard
 */

public class PaxCreator {

   private static final double[][] distributionSpeedValues = { //Lower,
Upper, Mode
        {0.8, 1.2, 1},
        {1,1.5,1.25}
  };

    private  static  final  double[][]  distributionStorageValues  =  {
//Lower, Upper, Mode
        {5, 10, 7.5},
        {7.5,15,11.25}
  };


   public static PAX create(int id, int year, String ref) {
   GregorianCalendar cal = (GregorianCalendar)  Calendar.getInstance();
       int currentYear = cal.get(GregorianCalendar.YEAR);
       int age = currentYear - year;
       int type = (age<=65)?1:2; //Ternary if
```

```
        double speedLower = distributionSpeedValues[type-1][0];
        double speedUpper = distributionSpeedValues[type-1][1];
        double speedMode = distributionSpeedValues[type-1][2];

        double    speed    =    Utils.triangular(speedLower,    speedUpper,
speedMode);

        double storageLower = distributionStorageValues[type-1][0];
        double storageUpper = distributionStorageValues[type-1][1];
        double storageMode = distributionStorageValues[type-1][2];

        double  storage  =  Utils.triangular(storageLower,  storageUpper,
storageMode);

        return new PAX(id, year,ref, type,age, speed, storage);
    }
}
```

- **Seat.java**

```
package abp_solgen;

import java.util.LinkedList;

/**
 *
 * @author Gerard
 */
public class Seat {

    private final int row;
    private final int seat;

    public Seat(int row, int seat) {
        this.row = row;
        this.seat = seat;
    }

      public static LinkedList<Seat> generateSeats(int rows, int
      seats) {
        LinkedList<Seat> seatList = new LinkedList<Seat>() ;
        for(int i = 1; i <= rows; i++) {
            for(int j = 1; j <= seats;j++) {
                Seat seat = new Seat(i, j);
                seatList.add(seat);
            }
        }
        return seatList;
    }

 }
```

- **Simulator.java**

```
package abp_solgen;

import java.util.LinkedList;
import java.util.PriorityQueue;

import javax.swing.JFrame;

import visualsim.FlyRepr;

import abp_solgen.api.BoardingStrategy;
import abp_solgen.api.Representation;
import abp_solgen.utils.Utils;

/**
 *
 * @author Gerard
 */

/**
   Simulator class is used to simulate the aircraft boarding
   process using the seat allocation and boarding groups created
  by the algorithm to estimate the total boarding time.
 */

public class Simulator {


    private final Input input;
    private final BoardingStrategy strategy;
    private double result = 0;
    private Representation f;
    private final boolean demoMode;
    private int seatInterferences = 0;

    public  Simulator(Input  input,  BoardingStrategy  strategy,
     boolean demoMode) {
        this.input = input;
        this.strategy = strategy;

        this.demoMode = demoMode;
        if(demoMode) {
            this.f = FlyRepr.getInstance();
        }
    }

    public static final int AISLE = 3;

      //enum defines a new EventStatus variable with a set of
      predefined constants. In this case: walking, storing and
      sitting.
```

```
    public enum EventStatus {
       WALKING, STORING, INTERFERENCE, SITTING
    }


private      static      final      double      types[][]      =
{{2.0,3.0,2.2},{4.0,5.0,4.2}}; //types array represent the time
penalty caused by a seat interference depending on the passenger
type

public static double getInterferenceTimeByType(int type) {
        return   Utils.triangular(types[type-1][0],   types[type-
        1][1], types[type-1][2]);
    }


    public void run() {
        double accumTime = 0;

        int test = 0;
        double[][] seats = new double[31][7];


       Event[][] aisleEvents = new Event[31][7]; //aisleEvents
      is an event matrix

        PriorityQueue<Event>          queue          =          new
        PriorityQueue<Event>();//queue represents the event list

        strategy.fill(queue,  input);  //prepare  passengers  for
        simulations
        LinkedList<Event> curr = new LinkedList<>();
        while(!queue.isEmpty()) curr.add(queue.poll());

        for(Event e: curr) {
            queue.add(e);
        }
        double currentTime = 0;
        double lastTime = 0;

      Event last = null;

       int iter = 0;
        while(!queue.isEmpty()) { //While the event list is not
        empty:
            iter++;
            boolean enqueue = false;
            Event currentEvent = queue.poll();//The current event
            (currentEvent) is the event at the top of the list
            currentTime  =  currentEvent.getTime();  //The  current
            time (currentTime) is the current event's time
            PAX pax = currentEvent.getPax();

            if(currentEvent.getStatus() == EventStatus.WALKING) {
            //If the passenger of the current event is walking:
                enqueue = true;
```

```
if(currentEvent.getLastRow() != pax.getRow()) { //If the
passenger hasn't reched his/her assigned row number:

int nextRow = currentEvent.getLastRow()+1; //The next row is the
current row+1

Event eventInProgress = aisleEvents[nextRow][AISLE]; //The event
in progress corresponds to the cell in front of the passenger's
location

if(eventInProgress == null) { //If there is not any event in the
current event next position:

aisleEvents[currentEvent.getLastRow()][AISLE] = null; //The
current event position is released

aisleEvents[nextRow][AISLE] = currentEvent; //The current event
is set in the event in progress position

currentEvent.setLastRow(nextRow); //The passenger's last row is
updated

currentEvent.incTime(pax.getSpeed()); //The event time is
increased
                    } else { //Else, the passenger has to wait
until the eventInProgress is completed


currentEvent.setTime(eventInProgress.getTime()+0.1);
                        last = currentEvent;
                    }
                }

else { //If the passenger has reached his/her assigned row;
he/she can proceed to store the hand luggage

currentEvent.setStatus(EventStatus.STORING);
//
FR.write(currentEvent.getPax().getPAX_ID()+"\tARRIVED\t"+current
Event.getTime()+"\tCurrRS("+currentEvent.getLastRow()+","+curren
tEvent.getLastSeatPos()+")"+"                                   dest
"+currentEvent.getPax().getRow()+","+currentEvent.getPax().getSe
at());

currentEvent.incTime(pax.getStorage_Speed());
                }
            }

else if(currentEvent.getStatus() == EventStatus.STORING) { //If
the passenger of the current event is storing his/her hand
luggage:
//
FR.write(currentEvent.getPax().getPAX_ID()+"\tSTORED\t"+currentE
vent.getTime()+"\tCurrRS("+currentEvent.getLastRow()+","+current
Event.getLastSeatPos()+")"+"                                    dest
"+currentEvent.getPax().getRow()+","+currentEvent.getPax().getSe
at());
```

```
enqueue = true;
int step = (pax.getSeat() > 3)? 1: -1;
int currentSeatPos = currentEvent.getLastSeatPos();

                int currRow = pax.getRow();
                double penalty = 0;
                int i = currentSeatPos;
                while(i != pax.getSeat()) {
                    i += step;
                    penalty                                          +=
seats[currentEvent.getLastRow()][i];
                }

currentEvent.setStatus(EventStatus.INTERFERENCE);
                currentEvent.incTime(penalty);
                if(penalty > 0) {
                    seatInterferences++;
                    test++;

                }

//
            }       else       if(currentEvent.getStatus()       ==
EventStatus.INTERFERENCE) {
//
FR.write(currentEvent.getPax().getPAX_ID()+"\tEND_INTERFERENCED\
t"+currentEvent.getTime()+"\tCurrRS("+currentEvent.getLastRow()+
","+currentEvent.getLastSeatPos()+")"+"                          dest
"+currentEvent.getPax().getRow()+","+currentEvent.getPax().getSe
at());

enqueue = true;
int currentSeatPos = currentEvent.getLastSeatPos();
int currRow = pax.getRow();
aisleEvents[currRow][currentSeatPos] = null;
currentEvent.setLastSeatPos(pax.getSeat());
currentEvent.setStatus(EventStatus.SITTING);

            }
 else if(currentEvent.getStatus() == EventStatus.SITTING) {
//
FR.write(currentEvent.getPax().getPAX_ID()+"\tSITTED\t"+currentE
vent.getTime()+"\tCurrRS("+currentEvent.getLastRow()+","+current
Event.getLastSeatPos()+")"+"                                     dest
"+currentEvent.getPax().getRow()+","+currentEvent.getPax().getSe
at());

seats[currentEvent.getLastRow()][currentEvent.getLastSeatPos()]
= getInterferenceTimeByType(pax.getType());

aisleEvents[currentEvent.getLastRow()][currentEvent.getLastSeatP
os()] = null;
            }
```

```
      //if not finished....
                lastTime = Math.max(lastTime, currentEvent.getTime());
                  if(enqueue) {
                      queue.add(currentEvent);
                  }

                  if(demoMode) {
                        f.present(curr);
                        f.redraw(currentTime);
                  }
               }
               accumTime = lastTime;
              if(demoMode) f.redraw(accumTime);
               result = accumTime;
          }


            public double getResult() {
                  return result;
             }

           public int getSeatInterferences() {
               return seatInterferences;
           }
      }
```

- **Solution.java**

```
package abp_solgen;

public class Solution {

    private final int nIter;

    private final double cost;

    private final int interferences;


    private Solution(int nIter, double cost, int interferences) {

        this.nIter = nIter;

        this.cost = cost;

        this.interferences = interferences;

    }


    public double getCost() {

        return cost;

    }
```

```
    public int getInterferences() {
        return interferences;
    }

public static Solution newSol(int nIter, double cost, int interferences) {
        return new Solution(nIter, cost, interferences);
    }

    @Override
public String toString() {
return "[Boarding time =" + cost + "\t seat interferences=" + interferences + ']';
    }
}
```

- **Utils.java**

```
package abp_solgen.utils;

import java.util.LinkedList;
import java.util.Random;
public class Utils {
private static Random rng;
public static void setSeed(long seed) {
      rng = new Random(seed);
    }
 public static double triangular(double a, double b, double c) {
        double U = rng.nextDouble();
        double F = (c-a) / (b-a);
        if(U <= F) return a + Math.sqrt(U * (b-a) * (c-a));
        else return b - Math.sqrt((1-U) * (b-a) * (b - c));
    }
```

```
    public static void ListShuffle(LinkedList list) {
        for(int i = 1; i < list.size(); i++) {
            int nPos = rng.nextInt(i);
            Object tmp = list.remove(i);
            Object curr = list.remove(nPos);
            list.add(nPos, tmp);
            list.add(i, curr);
        }
    }
    public static void main(String[] args) {
        LinkedList<Integer> aaa = new LinkedList<Integer>();
        setSeed(89);
        for(int i = 0; i < 100; i++) {
            aaa.add(i);
        }
        for(int i : aaa) {
            System.out.print(i+" ");
        }
        System.out.println();
        ListShuffle(aaa);
        for(int i : aaa) {
            System.out.print(i+" ");
        }
    }
    public static boolean inRange(int val, int min, int max) {
        return val >= min && val <= max;
    }
}
```

- **ABPStrategy.java**

```
package abp_solgen.abpstrategy;

import abp_solgen.Event;
import abp_solgen.Input;
import abp_solgen.PAX;
import abp_solgen.PAXGroup;
import abp_solgen.Simulator;
import abp_solgen.api.BoardingStrategy;
import abp_solgen.utils.Utils;
import java.util.Collections;
import java.util.Comparator;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.PriorityQueue;

/**
 *
 * @author Gerard
 */
public class ABPStrategy implements BoardingStrategy {

    private ABPStrategy() {
    }

    public static ABPStrategy getStrategy() {
        return new ABPStrategy();
    }

    @Override
    public void assignBoardingGroup(Input input) {


//Assign seats and boarding groups to passengers travelling in
groups of more than two passengers

        AssignGroups as = new AssignGroups(iRow, iCol);
        as.prepareGroups(input.getGroups()); //The groups are sorted by
      ascending size order

        for(PAXGroup p: input.getGroups()) {
            for(PAX pax: p.getGroup()) {
                as.apply(pax);
            }
            as.assignBoardingGroup(p);
            input.registerGroup(p);
        }


 //Assign seats and boarding groups to passengers pairs

        AssignPairs ap = new AssignPairs(as.getRow()-1, iCol);
        ap.prepareGroups(input.getPairs()); //The pairs are sorted by
      passenger type
        for(PAXGroup p: input.getPairs()) {
            for(PAX pax: p.getGroup()) {
                ap.apply(pax);
            }
            ap.assignBoardingGroup(p);
            input.registerGroup(p);
        }
```

```
    // Assign seats and boarding groups to passengers travelling
    alone

      AssignAlone aa2 = new AssignAlone(ap.getRow()-1, iCol);
     aa2.prepareGroups(input.getAlone()); //Passengers are sorted by
    type

     Iterator<PAXGroup> iterator = input.getAlone().iterator(); //An
    iterator object is declared to run over the alone passenger list

    //While there are free seats before reaching the top of the
    aircraft and there are passengers left to be assigned,passengers
    are assigned seats according to the ABP_SolGen policy.

     while(aa2.hasNextPos() && iterator.hasNext()) {
          PAXGroup p = iterator.next();
          PAX pax = p.getFirst();
          aa2.apply(pax);
          aa2.assignBoardingGroup(p);
          input.registerGroup(p);
      }

    //While there are passengers left to be assigned and there are
    free seats in the groups section, alone passengers are assigned
    to the free seats.

      while(iterator.hasNext() && as.getAvailableSeats() > 0) {
            PAXGroup p = iterator.next();
            PAX pax = p.getFirst();
            as.apply(pax);
            as.assignBoardingGroup(p);
            input.registerGroup(p);
      }

    //While there are passengers left to be assigned and there are
    free seats in the pairs section, alone passengers are assigned to
    the free seats.

      while(iterator.hasNext() && ap.getAvailableSeats() > 0) {
            PAXGroup p = iterator.next();
            PAX pax = p.getFirst();
            ap.apply(pax);
            ap.assignBoardingGroup(p);
            input.registerGroup(p);
      }
}

public static final Comparator<PAX> ASCENDENT_SEAT_ORDER = new
Comparator<PAX>() {

    @Override
    public int compare(PAX o1, PAX o2) {
       int n = o1.getSeat();
       if(n > 3)
            return o2.getSeat() - o1.getSeat();
       else
            return o1.getSeat() - o2.getSeat();
    }

};
```

```
        @Override
        public void fill(PriorityQueue<Event> queue, Input input) {

    //Iterate over boarding groups

    for(Integer boardingGroupCode: input.getBoardingGroups().keySet()) {
                LinkedList<PAXGroup>          currentBoardingGroup          =
                input.getBoardingGroups().get(boardingGroupCode);

                 //Shuffle groups in the boarding group
                 Utils.ListShuffle(currentBoardingGroup);

                 //Iterate over individual groups
                for(PAXGroup group: currentBoardingGroup) {

                    //Shuffle PAX in each group
                    if(group.getGroup().size() == 2) {

                        Collections.sort(group.getGroup(),ASCENDENT_SEAT_ORD
                        ER );
                    }

                  else {
                        Utils.ListShuffle(group.getGroup());
                    }
                    for(PAX pax: group.getGroup()) {
                        //System.out.println(pax);
                        queue.add(new Event(pax));
                    }
                }

            }
        }

    }
```

- **AssignAlone.java**

```
package abp_solgen.abpstrategy;


import abp_solgen.PAX;

import abp_solgen.PAXGroup;

import java.util.Collections;

import java.util.LinkedList;


/**

    AssignAlone class assigns seats to passengers travelling alone according
to the ABP_SolGen heuristic criteria.

 */
```

```
public class AssignAlone {


      private int[] sequence = {0,6,1,5,2,4};//Defines the seat allocation
      sequence.

      private int row; //Row number.

      private int currCol; //Current seat column.

      private final int startingRow; //Represents the first row to be assigned
      to passengers traveling alone.


    /**

        AssignAlone constructor sets the row and current seat column.

        @param row is the row number where the seat allocation is started.

        @param col is the seat to be assigned in the first place.

     */

    public AssignAlone(int row, int col) {

        startingRow = row;

        this.row = row;

        this.currCol = col;

    }


    /**

        apply method assigns a seat to a passenger.

        @param p is the passenger to be allocated.

     */

    public void apply(PAX p) {

       if(row == 0) {    //If there aren't more rows in the aircraft, the
current row is set to the starting row.

           currCol++;

           row = startingRow;

       }

       p.setSeat(row, sequence[currCol]);

       row--;


    }
```

```
    public boolean hasNextPos() {

        return !(currCol == sequence.length -1 && row == 0);

    }


    /**

       prepareGroups method sorts a list of passenger groups by ascending
       passenger type order.

       @param groups contains a list of passenger groups.

     */

    public void prepareGroups(LinkedList<PAXGroup> groups) {

        Collections.sort(groups, PAXGroup.TYPE_SORT);

    }


   /**

       assignBoardingGroup assigns a boarding group to a passenger depending on
       the assigned seat.

      @param pGroup represents the set of passengers to be assigned.

    */

   public void assignBoardingGroup(PAXGroup pGroup){

        PAX p = pGroup.getFirst();

        final int[] gRow = {2,3,6,0,7,5,4};


        pGroup.setBoardingGroup(gRow[p.getSeat()]);

    }



    public int getRow() {

        return row;

    }

}
```

- **AssignGroups.java**

```
package abp_solgen.abpstrategy;


import abp_solgen.PAX;

import abp_solgen.PAXGroup;

import java.util.Collections;

import java.util.LinkedList;

/**

    AssignGroups class assigns seats to passengers travelling in groups of
more than 2 members according to the ABP_SolGen heuristic criteria.

 */


public class AssignGroups {

public static final int BIG_GROUP = 1;


private int[] sequence = {0,1,2,6,5,4};//Defines the seat allocation sequence.

private int row; //Row number.

private int currCol; //Current seat column.


    /**

        AssignGroups constructor sets the row and current seat column.

        @param row is the row number where the seat allocation is started.

        @param col is the seat to be assigned in the first place.

     */

    public AssignGroups(int row, int col) {

        this.row = row;

        this.currCol = col;

    }

    /**

        apply method assigns a seat to a passenger.

        @param p is the passenger to be allocated.

     */
```

```
public void apply(PAX p) {

   if(currCol == 6) {    //If the current row is full, assign the following
   row in descending order.

        currCol = 0;

        row--;

    }

    p.setSeat(row, sequence[currCol]);

    currCol++;


}


/**

   prepareGroups method sorts a list of passenger groups by ascending group
   size order.

     @param groups contains a list of passenger groups.

 */

public void prepareGroups(LinkedList<PAXGroup> groups) {

     Collections.sort(groups);

}

/**

   getAvailableSeats method is used to calculate the number of free seats
   left in the grups section of the aicraft cabin.

     @return returns the number of free seats left in the groups section.

 */

public int getAvailableSeats() {

     return sequence.length - currCol;

}


 public void assignBoardingGroup(PAXGroup pGroup){


     pGroup.setBoardingGroup(BIG_GROUP);


}
```

```
    public int getRow() {
        return row;
    }
}
```

- **AssignPairs.java**

```
package abp_solgen.abpstrategy;

import abp_solgen.PAX;
import abp_solgen.PAXGroup;
import java.util.Collections;
import java.util.LinkedList;

/**
    AssignPairs class assigns seats to passengers travelling in pairs according to the ABP_SolGen heuristic criteria.
 */
public class AssignPairs {

       private int[] sequence = {0,1,2,4,5,6}; //Defines the seat allocation sequence.
       private int row; //Row number.
       private int currCol; //Current seat column.

    /**
        AssignPairs constructor sets the row and current seat column.
        @param row is the row number where the seat allocation is started.
        @param col is the seat to be assigned in the first place.
     */
    public AssignPairs(int row, int col) {
        this.row = row;
        this.currCol = col;
    }
```

```
    /**
        apply method assigns a seat to a passenger.
        @param p is the passenger to be allocated.
     */
    public void apply(PAX p) {
       if(currCol == 6) { //If the current row is full, assign the following
row in descending order.
           currCol = 0;
           row--;
       }
       p.setSeat(row, sequence[currCol]);
       currCol++;
    }

    /**
        prepareGroups method sorts a list of passenger groups by ascending
passenger type order.
        @param groups contains a list of passenger groups.
     */
    public void prepareGroups(LinkedList<PAXGroup> groups) {
        Collections.sort(groups, PAXGroup.TYPE_SORT);
    }

    public int getAvailableSeats() {
        return sequence.length - currCol;
    }
    public void assignBoardingGroup(PAXGroup pGroup){
        PAX p = pGroup.getFirst();
        final int[] evenRows = {2,2,6,0,6,4,4};
        final int[] oddRows = {3,3,7,0,7,5,5};
        int group = 0;
        if(p.getRow()%2==0){
           group = evenRows[p.getSeat()];
        }
```

```
 else {
            group = oddRows[p.getSeat()];
        }
        pGroup.setBoardingGroup(group);
    }

    public int getRow() {
        return row;
    }
}
```

- **B2FStrategy.java**

```
package abp_solgen.backtofront;

import abp_solgen.Event;
import abp_solgen.Input;
import abp_solgen.PAX;
import abp_solgen.PAXGroup;
import abp_solgen.api.BoardingStrategy;
import static abp_solgen.api.BoardingStrategy.iCol;
import static abp_solgen.api.BoardingStrategy.iRow;
import abp_solgen.randomstrategy.RandomStrategy;
import abp_solgen.utils.Utils;
import java.util.Collections;
import java.util.Comparator;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.PriorityQueue;
```

```
public class B2FStrategy implements BoardingStrategy {

    public static final Comparator<PAX> DESCENDENT_ROW = new Comparator<PAX>()
{

        @Override

        public int compare(PAX o1, PAX o2) {

           return o2.getRow() - o1.getRow();

        }


    };


       public static BoardingStrategy getStrategy() {

        return new B2FStrategy();

    }


     @Override

    public void fill(PriorityQueue<Event> queue, Input input) {

        //Iterate over boarding groups

        LinkedList<PAX> preList = new LinkedList<PAX>();

        for(Integer boardingGroupCode: input.getBoardingGroups().keySet()) {

      LinkedList<PAXGroup>               currentBoardingGroup               =
      input.getBoardingGroups().get(boardingGroupCode);

        //Shuffle groups in the boarding group

             Utils.ListShuffle(currentBoardingGroup);

            //Iterate over individual groups

            for(PAXGroup group: currentBoardingGroup) {

                //Shuffle PAX in each group

                 Utils.ListShuffle(group.getGroup());


                for(PAX pax: group.getGroup()) {

                    preList.add(pax);

                }

            }

        }
```

```
    for(PAX pax: preList) {
         // System.out.println(pax);
          queue.add(new Event(pax));
      }

}

@Override
public void assignBoardingGroup(Input input) {

   //Assign seats and boarding groups to passengers travelling in groups of
   more than two passengers
   AssignGroups as = new AssignGroups(iRow, iCol);
   as.prepareGroups(input.getGroups());  //The  groups  are  sorted  by
   ascending size order
    for(PAXGroup p: input.getGroups()) {
        for(PAX pax: p.getGroup()) {
            as.apply(pax);
        }
        as.assignBoardingGroup(p);
        input.registerGroup(p);
    }

    //Assign seats and boarding groups to passengers pairs
    AssignPairs ap = new AssignPairs(as.getRow()-1, iCol);
   ap.prepareGroups(input.getPairs()); //The pairs are sorted by passenger
   type
    for(PAXGroup p: input.getPairs()) {
        for(PAX pax: p.getGroup()) {
            ap.apply(pax);
        }
        ap.assignBoardingGroup(p);
        input.registerGroup(p);
    }
```

```
  //Assign seats and boarding groups to passengers travelling alone

  AssignAlone aa2 = new AssignAlone(ap.getRow()-1, iCol);

  aa2.prepareGroups(input.getAlone()); //Passengers are sorted by type

Iterator<PAXGroup> iterator = input.getAlone().iterator(); //An iterator
object is declared to run over the alone passenger list


//While there are free seats before reaching the top of the aircraft and
there are passengers left to be assigned,

  //passengers are assigned seats according to the ABP_SolGen policy.

  while(aa2.hasNextPos() && iterator.hasNext()) {

      PAXGroup p = iterator.next();

      PAX pax = p.getFirst();

      aa2.apply(pax);

      aa2.assignBoardingGroup(p);

      input.registerGroup(p);

  }


//While there are passengers left to be assigned and there are free
seats in the groups section, alone passengers are

  //assigned to the free seats.

  while(iterator.hasNext() && as.getAvailableSeats() > 0) {

        PAXGroup p = iterator.next();

        PAX pax = p.getFirst();

        as.apply(pax);

        as.assignBoardingGroup(p);

        input.registerGroup(p);

  }


//While there are passengers left to be assigned and there are free
seats in the pairs section, alone passengers are

  //assigned to the free seats.

  while(iterator.hasNext() && ap.getAvailableSeats() > 0) {

        PAXGroup p = iterator.next();

        PAX pax = p.getFirst();
```

```
                ap.apply(pax);
                ap.assignBoardingGroup(p);
                input.registerGroup(p);
            }
        }
        public static int getBoardingGroup(int row) {
            int currentGroup = 0;
            if  (Utils.inRange(row, 1, 7))      currentGroup = 4;
            else if(Utils.inRange(row, 8, 14))  currentGroup = 3;
            else if(Utils.inRange(row, 15, 22)) currentGroup = 2;
            else if(Utils.inRange(row, 23, 30)) currentGroup = 1;

            return currentGroup;
        }
}
```

- **F2BStrategy.java**

```
package abp_solgen.fronttoback;

import abp_solgen.Event;
import abp_solgen.Input;
import abp_solgen.PAX;
import abp_solgen.PAXGroup;
import abp_solgen.api.BoardingStrategy;
import static abp_solgen.api.BoardingStrategy.iCol;
import static abp_solgen.api.BoardingStrategy.iRow;
import abp_solgen.randomstrategy.RandomStrategy;
import abp_solgen.utils.Utils;
import java.util.Collections;
import java.util.Comparator;
import java.util.Iterator;
import java.util.LinkedList;
```

```
import java.util.PriorityQueue;


public class F2BStrategy implements BoardingStrategy {

    public static final Comparator<PAX> DESCENDENT_ROW = new Comparator<PAX>()
    {

        @Override

        public int compare(PAX o1, PAX o2) {

           return o2.getRow() - o1.getRow();

        }


    };


       public static BoardingStrategy getStrategy() {

        return new F2BStrategy();

    }


     @Override public void fill(PriorityQueue<Event> queue, Input input) {

        //Iterate over boarding groups

        LinkedList<PAX> preList = new LinkedList<PAX>();

        for(Integer boardingGroupCode: input.getBoardingGroups().keySet()) {

      LinkedList<PAXGroup>                currentBoardingGroup                =
      input.getBoardingGroups().get(boardingGroupCode);

        //Shuffle groups in the boarding group

            Utils.ListShuffle(currentBoardingGroup);

            //Iterate over individual groups

            for(PAXGroup group: currentBoardingGroup) {

                //Shuffle PAX in each group

                Utils.ListShuffle(group.getGroup());


                for(PAX pax: group.getGroup()) {

                    preList.add(pax);

                }

            }

        }
```

```
    for(PAX pax: preList) {
         // System.out.println(pax);
          queue.add(new Event(pax));
      }
}
@Override
public void assignBoardingGroup(Input input) {

  //Assign seats and boarding groups to passengers travelling in groups of
  more than two passengers
    AssignGroups as = new AssignGroups(iRow, iCol);
  as.prepareGroups(input.getGroups());  //The  groups  are  sorted  by
  ascending size order
    for(PAXGroup p: input.getGroups()) {
        for(PAX pax: p.getGroup()) {
            as.apply(pax);
        }
        as.assignBoardingGroup(p);
        input.registerGroup(p);
    }

    //Assign seats and boarding groups to passengers pairs
    AssignPairs ap = new AssignPairs(as.getRow()-1, iCol);
  ap.prepareGroups(input.getPairs()); //The pairs are sorted by passenger
  type
    for(PAXGroup p: input.getPairs()) {
        for(PAX pax: p.getGroup()) {
            ap.apply(pax);
        }
        ap.assignBoardingGroup(p);
        input.registerGroup(p);
    }
```

```
  //Assign seats and boarding groups to passengers travelling alone

  AssignAlone aa2 = new AssignAlone(ap.getRow()-1, iCol);

  aa2.prepareGroups(input.getAlone()); //Passengers are sorted by type

Iterator<PAXGroup> iterator = input.getAlone().iterator(); //An iterator
object is declared to run over the alone passenger list


//While there are free seats before reaching the top of the aircraft and
there are passengers left to be assigned,

  //passengers are assigned seats according to the ABP_SolGen policy.

  while(aa2.hasNextPos() && iterator.hasNext()) {

      PAXGroup p = iterator.next();

      PAX pax = p.getFirst();

      aa2.apply(pax);

      aa2.assignBoardingGroup(p);

      input.registerGroup(p);

  }


//While there are passengers left to be assigned and there are free
seats in the groups section, alone passengers are

  //assigned to the free seats.

  while(iterator.hasNext() && as.getAvailableSeats() > 0) {

        PAXGroup p = iterator.next();

        PAX pax = p.getFirst();

        as.apply(pax);

        as.assignBoardingGroup(p);

        input.registerGroup(p);

  }


//While there are passengers left to be assigned and there are free
seats in the pairs section, alone passengers are

  //assigned to the free seats.

  while(iterator.hasNext() && ap.getAvailableSeats() > 0) {

        PAXGroup p = iterator.next();

        PAX pax = p.getFirst();
```

```
                ap.apply(pax);
                ap.assignBoardingGroup(p);
                input.registerGroup(p);
            }
        }
        public static int getBoardingGroup(int row) {
            int currentGroup = 0;
            if  (Utils.inRange(row, 1, 7))      currentGroup = 1;
            else if(Utils.inRange(row, 8, 14))  currentGroup = 2;
            else if(Utils.inRange(row, 15, 22)) currentGroup = 3;
            else if(Utils.inRange(row, 23, 30)) currentGroup = 4;

            return currentGroup;
        }
}
```

- **RandomStrategy.java**

```
package abp_solgen.randomstrategy;

import abp_solgen.Event;
import abp_solgen.Input;
import abp_solgen.PAX;
import abp_solgen.PAXGroup;
import abp_solgen.Simulator;
import abp_solgen.api.BoardingStrategy;
import abp_solgen.utils.Utils;
import java.util.Collections;
import java.util.Comparator;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.PriorityQueue;
```

```
/**
 *
 * @author Gerard
 */
public class RandomStrategy implements BoardingStrategy {

    protected RandomStrategy() {
    }

    public static BoardingStrategy getStrategy() {
        return new RandomStrategy();
    }

    @Override
    public void assignBoardingGroup(Input input) {


      //Assign seats and boarding groups to passengers travelling in groups of
      more than two passengers
       AssignGroups as = new AssignGroups(iRow, iCol);
      as.prepareGroups(input.getGroups());   //The   groups   are   sorted   by
      ascending size order
       for(PAXGroup p: input.getGroups()) {
           for(PAX pax: p.getGroup()) {
               as.apply(pax);
           }
           as.assignBoardingGroup(p);
           input.registerGroup(p);
       }
       //Assign seats and boarding groups to passengers pairs
       AssignPairs ap = new AssignPairs(as.getRow()-1, iCol);
      ap.prepareGroups(input.getPairs()); //The pairs are sorted by passenger
      type
```

```
  for(PAXGroup p: input.getPairs()) {
      for(PAX pax: p.getGroup()) {
          ap.apply(pax);
      }
      ap.assignBoardingGroup(p);
      input.registerGroup(p);
  }

  //Assign seats and boarding groups to passengers travelling alone
  AssignAlone aa2 = new AssignAlone(ap.getRow()-1, iCol);
  aa2.prepareGroups(input.getAlone()); //Passengers are sorted by type
Iterator<PAXGroup> iterator = input.getAlone().iterator(); //An iterator
object is declared to run over the alone passenger list

//While there are free seats before reaching the top of the aircraft and
there are passengers left to be assigned,
  //passengers are assigned seats according to the ABP_SolGen policy.
  while(aa2.hasNextPos() && iterator.hasNext()) {
      PAXGroup p = iterator.next();
      PAX pax = p.getFirst();
      aa2.apply(pax);
      aa2.assignBoardingGroup(p);
      input.registerGroup(p);
  }
//While there are passengers left to be assigned and there are free
seats in the groups section, alone passengers are
  //assigned to the free seats.
  while(iterator.hasNext() && as.getAvailableSeats() > 0) {
        PAXGroup p = iterator.next();
        PAX pax = p.getFirst();
        as.apply(pax);
        as.assignBoardingGroup(p);
        input.registerGroup(p);
  }
```

```
    //While there are passengers left to be assigned and there are free
    seats in the pairs section, alone passengers are

      //assigned to the free seats.

      while(iterator.hasNext() && ap.getAvailableSeats() > 0) {

            PAXGroup p = iterator.next();

            PAX pax = p.getFirst();

            ap.apply(pax);

            ap.assignBoardingGroup(p);

            input.registerGroup(p);

      }

}


public   static   final   Comparator<PAX>   ASCENDENT_SEAT_ORDER   =   new
Comparator<PAX>() {


      @Override

      public int compare(PAX o1, PAX o2) {

         return o1.getSeat() - o2.getSeat();

      }


};

@Override

public void fill(PriorityQueue<Event> queue, Input input) {

      //Iterate over boarding groups

      for(Integer boardingGroupCode: input.getBoardingGroups().keySet()) {

           LinkedList<PAXGroup>            currentBoardingGroup            =
           input.getBoardingGroups().get(boardingGroupCode);

      //Shuffle groups in the boarding group

          Utils.ListShuffle(currentBoardingGroup);

          Utils.ListShuffle(currentBoardingGroup);

          //Iterate over individual groups

          for(PAXGroup group: currentBoardingGroup) {

              //Shuffle PAX in each group

              //System.out.println(group.getBookRef());
```

```
                for(PAX pax: group.getGroup()) {
                  //  System.out.println(pax);
                    queue.add(new Event(pax));
                }
            }
        }
    }
}
```

- **BoardingStrategy.java**

```
package abp_solgen.api;

import abp_solgen.Event;
import abp_solgen.Input;
import java.util.PriorityQueue;
public interface BoardingStrategy {
    final int iRow = 30; //The starting row number is set to 30
    final int iCol = 0; //The starting column number is set to 0
    public void assignBoardingGroup(Input input);
    public void fill(PriorityQueue<Event> queue, Input input);
}
```

- **Representation.java**

```
package abp_solgen.api;
import java.util.LinkedList;
import abp_solgen.Event

public interface Representation {
       public void present(LinkedList<Event> events);
       public void redraw(double currentTime);
}
```