Universitat Autònoma de Barcelona

escola d'enginyeria

DETERMINACIÓN DE ACTIVIDADES CRÍTICAS EN LA ESCALA DE AERONAVES

Memòria del Treball Fi de Grau
Gestió Aeronàutica
realitzat per
Andrés San Antonio Bou
i dirigit per
Dr. Ángel A. Juan Sabadell,
a 7 de Juliol de 2014

El sotasignat, Ángel A. Juan

Professor/a de l'Escola d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en Andrés San Antonio Bou.

I per tal que consti firma la present.

Signat: .....

Sabadell, 7 de Juliol de 2014

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

Títol del projecte: Determinación de actividades críticas en la escala de aeronaves	
Autor[a]: Andrés San Antonio Bou	Data: Julio de 2014
Tutor[a]/s[es]: Ángel A. Juan	
Titulació: Grau en gestió Aeronàutica	
Paraules clau (mínim 3) • Català: Escala d'aeronaus, Coll d'ampolla, algoritme, gestió de recursos. • Castellà: Escala de aeronaves, Cuellos de botella, algoritmo, gestión de recursos. • Anglès: Aircraft turnaround, bottleneck, algorithm, Resources management.	
Resum del projecte (extensió màxima 100 paraules) • Català: En aquest projecte es presenta una eina destinada al suport de la gestió dels recursos en l'escala d'aeronaus. Es presenta una eina capaç de detectar aquelles activitats o grups d'activitats que creen colls d'ampolla en el procés. L'eina es capaç de detectar els colls d'ampolla aplicant temps estocàstics a les activitats associades a una distribució estadística de tipus *Weibull*. • Castellà: En este proyecto se presenta una herramienta destinada al soporte de la gestión de los recursos en la escala de aeronaves. Se presenta una herramienta capaz detectar aquellas actividades o grupos de actividades que crean cuellos de botella en el proceso. La herramienta es capaz de detectar los cuellos de botella aplicando tiempos estocasticos a las actividades asociados a una distribución estadística de tipo *Weibull*. • Anglès: In this project a tool to support the management of resources in aircraft turnaround is presented. A tool to identify those activities or groups of activities that create bottlenecks in the process. The tool is able to detect the bottlenecks using stochastic times for activities, associated with a statistical distribution of *Weibull* type.	

Índice

Sección 4: Experimentos computacionales

Sección 5: Conclusiones y trabajo futuro

Anexos

0 RESUMEN

El objetivo de este proyecto es crear una herramienta que permita detectar las actividades o grupos de actividades que forman cuellos de botella en la escala de aeronaves.

Utilizando tiempos variables obtenidos mediante distribuciones estadísticas, el algoritmo presentado permite detectar qué series de actividades son las que requieren más tiempo y las que menos. Es decir, aquellas actividades que son críticas para el cumplimiento esperado de la escala y aquellas que no representan un elevado riesgo a la hora de completar la escala esperada.

Con unos tiempos medios de duración y varianza conocidas, asociados a una distribución estadística, el algoritmo calcula diferentes tiempos para cada actividad en cada una de las iteraciones de la simulación. El algoritmo crea un fichero de texto en el que se muestran los resultados de la simulación y permite al usuario analizar la escala.

Esta herramienta permite al usuario detectar aquellas actividades en las que debe centrar sus esfuerzos para que se reduzca el tiempo total de la escala. También permite, al conocer las actividades que acaban antes, reasignar aquellos recursos que al acabar más pronto queden ociosos. Por tanto, es una buena herramienta de planificación y gestión de la escala de aeronaves.

De los experimentos y pruebas realizadas, se concluye que en las escalas de aeronaves de transporte de pasajeros, las actividades que están relacionadas directamente con los pasajeros son las que forman los cuellos de botella. Se define también una mejora de la escala del Boeing 737-800.

Por último, se exponen las conclusiones extraídas de todo el trabajo, las posibles mejoras y líneas a seguir en trabajos futuros.

Sección 1: Introducción

1 INTRODUCCIÓN Y MOTIVACIÓN

Actualmente el mercado de la aviación comercial se encuentra en un panorama altamente competitivo en el cual resulta necesario para las aerolíneas obtener el máximo beneficio posible. Este panorama no se debe a una situación puntual, sino que las expectativas de tráfico de pasajeros ofrecidas por Boeing (*www.boeing.com*) i Airbus (*www.airbus.com*) indican que el número de pasajeros crecerá alrededor de un 5% anual durante los 20 años siguientes.

Esta situación de expansión provocará un déficit de capacidad en los aeropuertos. Para mejorar esta capacidad futura, se han de buscar métodos de optimización de las infraestructuras y servicios existentes en la actualidad, minimizando al máximo posible la inversión en infraestructuras.

Uno de los considerados procesos críticos para las compañías, es el proceso de escala de las aeronaves (*turnaround*). La escala de una aeronave comprende todas las actividades que se han de desarrollar desde que el avión llega a la puerta de embarque o posición en remoto, hasta que se marcha de ella. Por ejemplo, embarque/desembarque de pasajeros, tareas de mantenimiento, carga y descarga de equipajes o carga, tareas de limpieza, etc.

Según la CODA (*Central Office of Delay Analysis*) de Eurocontrol, en el año 2013 los vuelos acumularon una media de 9.3 minutos de retraso por vuelo, de los cuales casi el 45% se deben a los retrasos denominados *Reactionary*, que son aquellos retrasos que se deben a un retraso en la llegada de la aeronave, de los pasajeros, de la tripulación o de la carga. Un 30% de los retrasos totales se atribuyen al tipo *Airline*, referentes a aquellos retrasos que son provocados por la aerolínea.

Este tipo de retrasos están relacionados, de forma directa, con el proceso de escala de aeronaves.

La escala de aeronaves es crítica debido a que si el tiempo que la aeronave está estacionada es superior al tiempo planificado, genera costes extra a la aerolínea e insatisfacción a los pasajeros **(Fricke & Schultz, 2009)**. Según un estudio realizado en la universidad de Westminster **(Cook et al., 2004)**, cada minuto de retraso tiene un coste medio de 25€.

Esto implica que el interés de las aerolíneas en minimizar esta ventana temporal sea muy alto. No únicamente provoca un elevado interés a las aerolíneas, también a los gestores aeroportuarios, ya que permite dotar de más capacidad a las infraestructuras aeroportuarias.

Uno de los rasgos característicos de este proceso se encuentra en que mientras algunas actividades únicamente se pueden desarrollar secuencialmente (no se puede comenzar a embarcar los pasajeros si no han desembarcado los del vuelo anterior), existen ciertas tareas que se pueden desarrollar paralelamente (a la vez que se desembarca el pasaje también se pueden descargar equipajes y carga).

El problema reside en que en las actividades a desarrollar no son completamente controlables, lo que provoca que exista un cierto grado de incertidumbre en el desarrollo de la actividad.

La motivación de este trabajo es la de ofrecer una herramienta que sea capaz de estimar cuáles son las actividades críticas más probables. Aplicando unos tiempos variables, calculados mediante distribuciones estadísticas, que se adecuen lo máximo posible a la realidad. Ofreciendo la ventaja que esta herramienta se puede utilizar para cualquier tipo de aeronave y cualquier tipo de escala siempre que se estructure correctamente la información en los *inputs* de la herramienta.

Conociendo las actividades críticas, se facilitan al gestor las tareas de planificación y reacción a los problemas. El conocimiento del comportamiento de las diferentes actividades soporta todas las actividades de gestión ofreciendo la posibilidad de optimizar el proceso.

2 OBJETIVOS

Se definen tres tipos de objetivos para la realización del trabajo: Objetivo Global, Objetivos básicos y Objetivos temporales.

2.1 Objetivo Global

El objetivo final del trabajo es ofrecer una herramienta sólida y eficaz, capaz de detectar las actividades que forman cuellos de botella en la escala de aeronaves, la cual permita al gestor tener una herramienta de soporte de decisiones para la reasignación de recursos con la finalidad de optimizar el proceso.

2.2 Objetivos básicos

- Recopilar y analizar la literatura existente sobre la optimización del *turnaround* con el objetivo de determinar las limitaciones o posibles mejoras de dichos estudios.
- Identificar las actividades que forman parte del proceso del *turnaround* con el objetivo de definir de una forma precisa el problema a resolver.
- La creación de un algoritmo capaz de determinar los caminos críticos del proceso de *turnaround* asignando a cada una de las actividades un tiempo estocástico.
- Determinar cuáles son las actividades que resultan críticas para la realización satisfactoria, en términos de tiempo, de todo el proceso de *turnaround*. Teniendo en cuenta que cada actividad tiene un tiempo de proceso variable.
- Realización de pruebas que permitan comprobar el correcto funcionamiento y que ayuden a explicar los beneficios del algoritmo creado.

2.3 Objetivos Complementarios

Si se dispone de tiempo, la creación del modelo en el programa de simulación SIMIO con el objetivo de hacer más visual el proceso y de comparar los resultados obtenidos en el modelo.

3 METODOLOGÍA

Una vez establecidos los objetivos, se ha de definir una metodología a seguir para el cumplimiento de los mismos. A continuación se definen los pasos a seguir:

1. Revisión de la literatura científica existente sobre el problema, con el objetivo de conocer los estudios realizados sobre el tema y comprender las diferentes metodologías y restricciones existentes. Este proceso permitirá conocer el estado del arte actual y ayudará a definir el problema formalmente.
2. Definición formal del problema a resolver de una forma formal e identificando de una forma clara las variables implicadas en el problema.
3. Creación de un algoritmo en leguaje de programación C++ con el objetivo de resolver el problema previamente definido.
4. Realización de simulaciones con el algoritmo definido en C++ con el objetivo de obtener datos para su posterior análisis.
5. Análisis de los resultados obtenidos gracias al algoritmo y definición de las conclusiones finales.
6. En caso de que sea factible, se creará un modelo sobre el problema con el software SIMIO con el objetivo de comparar los resultados obtenidos a la vez que ofrecer un modelo más visual que el definido con C++.

Sección 2: Conceptos básicos y estado actual

4 LA ESCALA DE AERONAVES

La escala de una aeronave (*turnaround*) es el proceso ordenado de actividades que se realizan desde que se ha estacionado la aeronave, hasta que la posición queda libre para otra aeronave. La finalidad es preparar la aeronave, la cual procede de otro aeropuerto, para el vuelo.

Las actividades, por su naturaleza y por el espacio físico en el que se realizan, pueden desarrollarse de forma secuencial o paralela. Por ejemplo, las actividades de desembarque y embarque de pasajeros deben ser secuenciales, mientras que la actividad de mantenimiento de la aeronave se puede realizar paralelamente a las dos anteriores, debido a que el espacio físico en el que se realizan no es el mismo.

En la Figura 1 se puede observar la escala propuesta por Airbus para el A320-200 para una escala de 45 minutos.

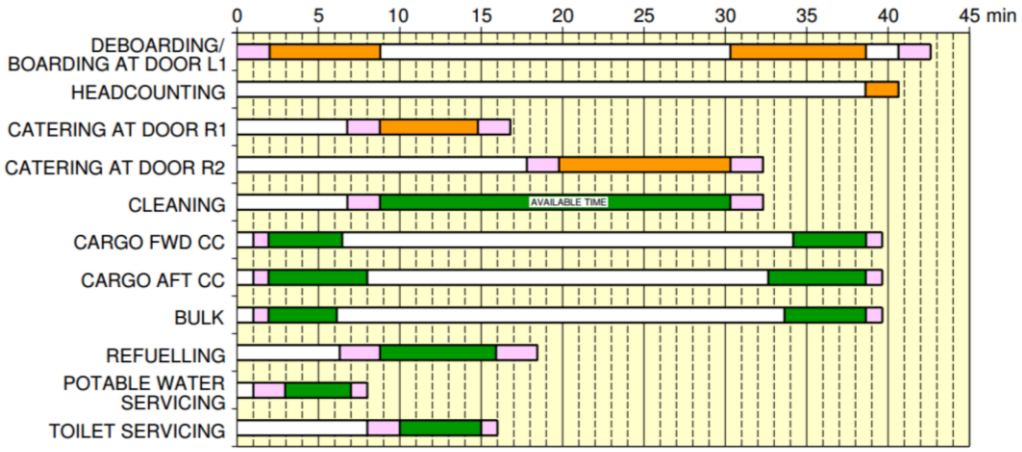


Figura 1 Escala comercial propuesta por Airbus para el A320-200 (AIRBUS S.A.S. 2005)

El tiempo de escala no es el mismo para todos los vuelos, principalmente depende de dos factores: La **longitud del vuelo** y el **tipo de vuelo**.

La longitud del vuelo es esencial a la hora de planificar el tiempo de escala. Si el vuelo es de largo recorrido la escala será mayor debido a que se suele tratar de aeronaves más grandes, lo que generalmente comporta unos tiempos mayores en todas las actividades. En caso contrario, si se trata de un vuelo de corto recorrido, los tiempos de desarrollo de las actividades serán significativamente menores.

El tipo de vuelo también es importante a la hora de planificar, ya que si se trata de un vuelo internacional los tiempos operativos del control de aduanas e inmigración son mayores que si se trata de un vuelo nacional o doméstico.

También hay que tener en cuenta el tipo de aerolínea. Si se trata de una aerolínea *low-cost* hay que tener en cuenta que generalmente este tipo de aerolíneas tienden a realizar únicamente las tareas estrictamente necesarias, ya que su principal objetivo es el de minimizar costes y por tanto minimizar la escala. Por ejemplo en ciertos vuelos *low-cost* es posible que se supriman las operaciones de catering.

Por otro lado están las compañías tradicionales, que buscan una mayor satisfacción del cliente y tienen una menor urgencia en este aspecto. En prácticamente todos los vuelos de estas compañías, por ejemplo, suele haber carga y descarga de catering.

4.1.1 Actividades básicas de la escala de aeronaves

A continuación se describen las actividades básicas de la escala de aeronaves:

- *Desembarque/Embarque de pasajeros y tripulación*

El embarque y desembarque de los pasajeros y de la tripulación se efectúa teniendo en cuenta dos factores principales:

En primer lugar, la **posición de la aeronave en el aeropuerto**, es decir, en caso de que el aeropuerto este situado en remoto, el desembarque y embarque se realizará mediante las escaleras de pasajeros y éstos se trasladaran a la terminal mediante autobuses o jardineras. Por otro lado, si la aeronave esta estacionada en una posición conectada a la terminal, estas tareas se realizarán mediante las pasarelas de conexión con la terminal.

En segundo lugar, la **estructura de la aeronave**. La forma de desembarcar y embarcar a los pasajeros depende de las salidas que tiene la aeronave, de los pasillos que tiene o de si se trata de una aeronave de dos plantas (B747 o A380).

Además, en el caso del embarque de pasajeros, una buena estrategia de embarque por parte de la aerolínea puede reducir considerablemente el tiempo de esta tarea.

Ilustración 1 Embarque de pasajeros en remoto

- *Descarga/Carga de equipaje y carga*

La carga y descarga del equipaje y la carga se puede realizar de dos formas diferentes, dependiendo del tipo de carga.

La carga se puede transportar mediante contenedores ULD o pallets, éstos permiten un manejo de la carga más rápido y una mejor organización de la misma dentro de la bodega de la aeronave. Por otro lado también se puede transportar la carga suelta (*bulk-load*), es decir, que cada unidad de carga se maneja individualmente. La *bulk-load* suele ser más ineficiente en términos de tiempo pero hay cierto tipo de carga que únicamente puede transportarse de esta forma.

Existe una gran cantidad de equipos que se pueden utilizar para realizar estas tareas pero los principales son los tractores y carros de equipaje, las cintas transportadoras para cargar el avión y las plataformas elevadoras para la carga de pallets y ULDs.

Ilustración 2 Descarga de ULDs utilizando plataformas elevadoras

- *Carga de Combustible*

La carga de combustible es un proceso que, aunque depende de la aeronave, suele realizarse de la misma forma para asegurar la estabilidad de la aeronave.

El proceso normal suele ser el siguiente: En primer lugar se llenan los tanques de combustible situados en las alas, asegurando así la estabilidad de la aeronave. Una vez se han llenado estos tanques, el resto se llenan en caso de que resulte necesario para el siguiente vuelo.

La carga de combustible se realiza con mangueras que se enganchan a la aeronave. Dependiendo de la infraestructura del aeropuerto, la fuente del combustible puede estar situada en una entrada directa en plataforma o en un camión cisterna.

Está permitida la carga de combustible mientras haya pasajeros en el interior de la aeronave, siempre que se cumplan ciertas condiciones, cómo por ejemplo que esté presente un equipo contraincendios en las inmediaciones de la operación o que estén todas las salidas libres para la evacuación en caso de emergencia y haya un tripulante en cabina.

Ilustración 3 Carga de combustible con toma de tierra

- *Mantenimiento rutinario*

El mantenimiento rutinario se entiende como aquellas tareas que, en la mayoría de los casos, se trata de la comprobación de los sistemas de la aeronave para asegurar el correcto funcionamiento de la misma durante el siguiente vuelo.

Algunas de las tareas más comunes son la comprobación del nivel de aceite de los motores, la revisión del líquido hidráulico, la de los niveles de oxígeno ola de la presión de las ruedas.

Las tareas concretas que se han de realizar en estas comprobaciones suelen ir definidas en los manuales de mantenimiento de cada avión. Estas tareas las pueden llevar a cabo tanto personal propio de la aerolínea como personal contratado.

También se pueden entender como tareas de mantenimiento aquellas que se refieren al cambio de agua potable y el drenado de los restos de los servicios.

- *Descarga/Carga de Catering*

La descarga y carga del catering consiste en el abastecimiento de alimentos y bebidas para los pasajeros.

Cómo se ha comentado previamente, este servicio suele ofrecerse sobre todo en vuelos de largo recorrido y en vuelos operados por aerolíneas tradicionales. Aunque también se puede realizar en vuelos *low-cost*, en los que se suele cobrar al pasaje una cierta cantidad de dinero a cambio de este servicio. Aun así, en servicios *low-cost* el tiempo dedicado a esta tarea suele ser menor que en el resto de vuelos.

La carga y descarga del catering se realiza utilizando plataformas elevadoras y los alimentos se transportan dentro del avión mediante carritos o *trolleys*. La carga del catering se suele realizar en el lado opuesto al que embarcan y desembarcan los pasajeros.

Ilustración 4 Operación de carga de catering

- *Limpieza de cabina*

La limpieza de la cabina se refiere a la limpieza general de la cabina de pasajeros tras un vuelo, con el objetivo de que la aeronave se encuentre en las condiciones higiénicas correctas para el siguiente vuelo.

Esta tarea suele ser llevada a cabo por personal propio de la compañía, por la misma tripulación o, en algunos casos, por parte de personal contratado. Esta última opción suele ser utilizada sobre todo por compañías tradicionales, mientras que las aerolíneas *low-cost* suele utilizar a la misma tripulación para la ejecución de estas tareas.

La limpieza de la cabina se lleva a cabo en el espacio temporal entre el desembarque y el embarque de los pasajeros, por lo que suele ser una actividad crítica.

- *Procedimientos de seguridad*

Los procedimientos de seguridad hacen referencia a ciertas comprobaciones que se han de llevar a cabo, antes de que embarquen los pasajeros, por parte de la tripulación dentro de la cabina.

El objetivo es asegurar el buen funcionamiento de sistemas orientados a preservar la seguridad de los pasajeros durante el vuelo en caso de emergencia.

Estos procedimientos son, por ejemplo, comprobar el sistema de inflado de las rampas de evacuación, las máscaras de oxígeno o los sistemas de prevención de incendios.

Esta tarea suele llevarse a cabo entre las tareas de limpieza y el embarque de pasajeros.

- *Lista de comprobación previa al vuelo*

La lista de comprobación previa al vuelo se refiere a la comprobación, por parte de la tripulación, del correcto funcionamiento de todos los sistemas de vuelo de la aeronave.

Esta tarea se lleva a cabo una vez se ha finalizado el embarque de los pasajeros y previamente a la salida de la aeronave de la posición de estacionamiento. Esta tarea se considera esencial para evitar posibles daños en la estructura de la aeronave durante el vuelo y para evitar posibles accidentes durante las operaciones del vuelo.

La lista de comprobación suele venir definida en el manual de la aeronave ya que no será la misma para todos los aviones.

5 REVISIÓN DE LA LITERATURA CIENTÍFICA

Sobre planificación, gestión y optimización del proceso de escala de aeronaves, se pueden distinguir dos formas de afrontar el problema:

Por una parte, se plantea el problema centrado en conocer el comportamiento de las actividades que forman todo el proceso. Este conocimiento facilita en gran medida las tareas de planificación y gestión del *turnaround*, por ejemplo, el conocimiento de las actividades que son críticas permitirá asignar a éstas más recursos y atenciones para así controlar y asegurar la buena realización de esa actividad.

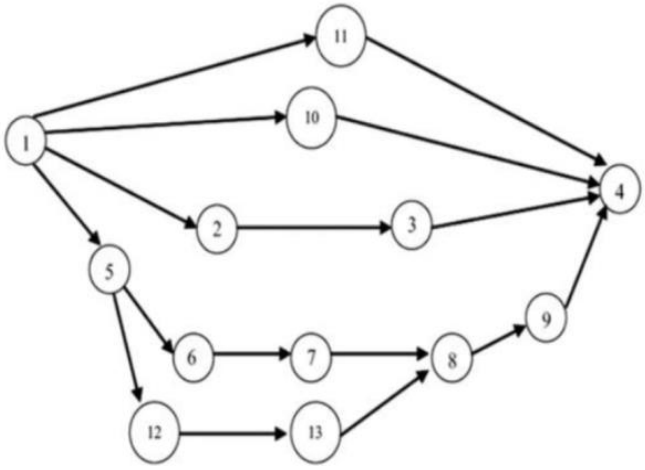
Por otra parte, se afronta el problema considerando la escala de aeronaves cómo un único bloque al cual se le asigna un buffer temporal al final de la escala que sea capaz de absorber los posibles retrasos que se hayan provocado durante el desarrollo del proceso.

5.1 Primer Problema: Planificación de actividades del *turnaround*

Cheng-Lung Wu (2010) define que la técnica más utilizada para la gestión de las operaciones terrestres de aerolíneas es la *Project Evaluation and Review Technique* (PERT) y la utilización de esta técnica tiene dos objetivos fundamentales. En primer lugar evaluar y mejorar la **eficiencia de los procesos** operacionales de la aerolínea. En segundo lugar evaluar y mejorar la eficiencia de la **asignación de recursos**, en especial, de los recursos humanos.

PERT se utiliza para representar procesos formados por diferentes actividades que tienen relaciones entre ellas y que además puede ser presentado en forma de red. En esta red los nodos representan las actividades y los arcos las dependencias que existen entre ellas.

La Figura 2, es la representación de las actividades de un *turnaround* estándar ofrecida por **C.L. Wu.** En ella, se definen 13 nodos (cada uno de ellos representa una actividad) y la relación entre sí (arcos) configurando esta forma de red. Se pueden observar relaciones secuenciales, p.ej. la actividad 7 no puede comenzar hasta que no haya terminado la 6, y paralelas, p.ej. las actividades 11 y 10 se pueden desarrollar paralelamente. La tabla inferior de la Figura 2 es una leyenda en la que están designados los tiempos de proceso de cada actividad.



Node	Activity	Duration
1	Aircraft on stand (start)	0
2	Cargo/baggage off-loading	20
3	Cargo/baggage loading	25
4	Push back/taxi (finish)	0
5	Passengers disembarking	10
6	Cabin cleaning	15
7	Security procedures	3
8	passenger boarding	15
9	Crew checklist procedures	3
10	Aircraft fuelling	25
11	Maintenance check	25
12	Catering off-loading	10
13	Catering loading	10

Figura 2 Representación en forma de red de las actividades del turnaround, Wu (2010)

Se pueden distinguir cinco caminos en la red, uno de los objetivos de éste planteamiento es el de determinar cuál de ellos es el **camino crítico**, es decir, aquel camino que implique más tiempo y que por tanto cualquier mínimo retraso en él se traduciría en un retraso global del *turnaround*. La determinación del camino crítico puede resultar de gran utilidad a la hora de planificar y gestionar el proceso.

El método para encontrar cuál es el camino crítico del proceso es el siguiente: Consiste, en primer lugar, en recorrer desde el primer nodo de la red hasta el último anotando el **comienzo más temprano** (ES) y la **finalización más temprana** (EC) de cada uno de los nodos teniendo en cuenta tiempos deterministas. Con este primer recorrido obtenemos el tiempo de finalización del *turnaround*. El segundo paso consiste en realizar el camino inverso nodo a nodo pero esta vez apuntando el **inicio más tardío** (LS) y la **finalización más tardía** (LC) teniendo en cuenta que el proceso no se puede retardar, es decir en el último nodo: $EC = LS = ES = LC$.

Una vez obtenido esto se puede calcular la holgura (*Slack*) que tiene cada una de las actividades calculando para cada nodo:

$$Slack_i = LC_i - EC_i = LS_i - ES_i$$

Con este cálculo obtenemos la holgura temporal de cada uno de los nodos, el camino crítico será aquel en el que todos los nodos tengan $Slack_i = 0$.

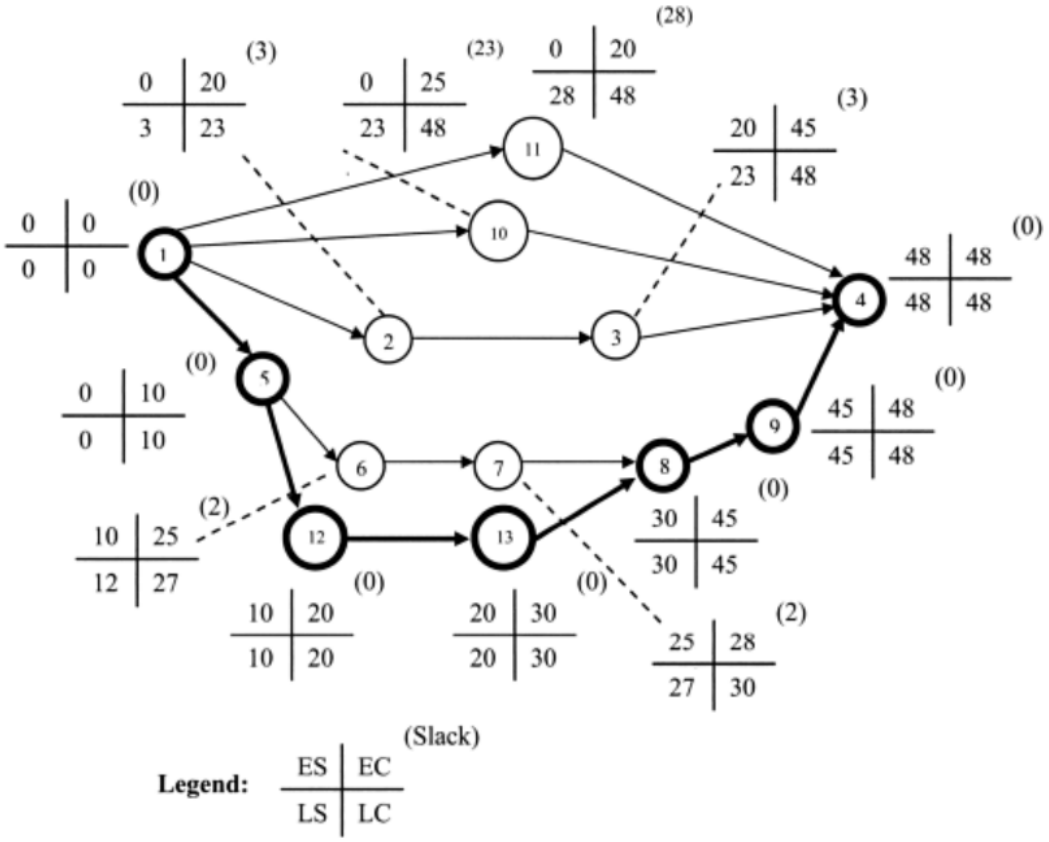


Figura 3 Ejemplo de determinación de camino crítico, Wu(2010)

Este proceso permite determinar el camino crítico, pero para aportarle más realidad al análisis se ha de añadir variabilidad en los tiempos de proceso. Para hacerlo, **C.L. Wu** define para cada actividad tres tiempos:

- **a**: tiempo de realización optimista.
- **b**: tiempo de realización pesimista.
- **m**: tiempo de realización más probable.

Utilizando una distribución Beta con estos tres parámetros para modelar el tiempo, es posible calcular la duración estimada de la actividad y su varianza. Una vez obtenemos las medias, gracias al Teorema del Límite Central, podemos obtener la probabilidad de que el tiempo esperado se cumpla.

En la Tabla 1 se puede observar un ejemplo en el que, además, se ha añadido un límite (*Milestone* (LC_i)) y se han calculado para cada una de las actividades las probabilidades de que el desarrollo de la actividad no supere el límite. Se puede observar que la probabilidad de que el tiempo total del *turnaround* no supere los 50 minutos es del 72%.

Activity	Longest Path	Mean	Variance	Milestone (LC_i)	Z_i	Prob($Z<z_i$)
2	(1–2)	20	2.78	23	1.80	96%
3	(1-2-3)	45	5.56	**48**	1.27	**90%**
5	(1-5)	10	2.78	10	0.00	50%
6	(1-5-6)	25	5.56	27	0.85	80%
7	(1-5-6-7)	28	5.67	**30**	0.84	**80%**
8	(1-5-12-13-8)	45	11.11	**48**	0.90	**82%**
9	(1-5-12-13-8-9)	48	11.22	48	0.00	50%
10	(1-10)	25	2.78	48	13.80	100%
11	(1-11)	20	2.78	48	16.80	100%
12	(1-5-12)	20	5.56	20	0.00	50%
13	(1-5-12-13)	30	8.33	**35**	1.73	**96%**
4	(1-5-12-13-8-9-4)	48	11.22	**50**	0.60	**72%**

Tabla 1 Probabilidad de cumplimiento los límites establecidos, Wu (2010)

Wu & Caves (2004) y **Cheng-Lung Wu (2010)** proponen una mejora al modelado del *turnaround* con PERT y CPM comentado previamente. Utilizando el método de la *cadena de Markov*.

La mejora que ofrece la utilización de la cadena de *Markov*, en comparación con los modelos de CPM, es que permite simular el comportamiento estocástico y dinámico entre las diferentes actividades del *turnaround* y modelar la ocurrencia de perturbaciones.

Mediante técnicas de Simulación Monte Carlo (SMC), el modelo permite dos aplicaciones básicas: **Simular la operación de una escala de aeronaves** y **optimizar el tiempo de escala.**

El modelo presentado se aplica para dos flujos secuenciales de actividades, los considerados críticos en el estudio: '*cargo and baggage handling*' y '*cabin cleaning and passenger processing*'.

Para cada uno de los flujos se definen los estados principales y los estados que representan los diferentes tipos de perturbaciones, asociados a cada uno de los estados principales. En la Figura 4 se muestra el flujo '*Cargo and baggage handling*' en el que los estados principales son el 1 (Llegada), 2 (Descarga), 3 (Carga), 4(Salida). Los estados que van del 5 al 9 representan las posibles razones de perturbación, tal y cómo se muestra en la Tabla 2.

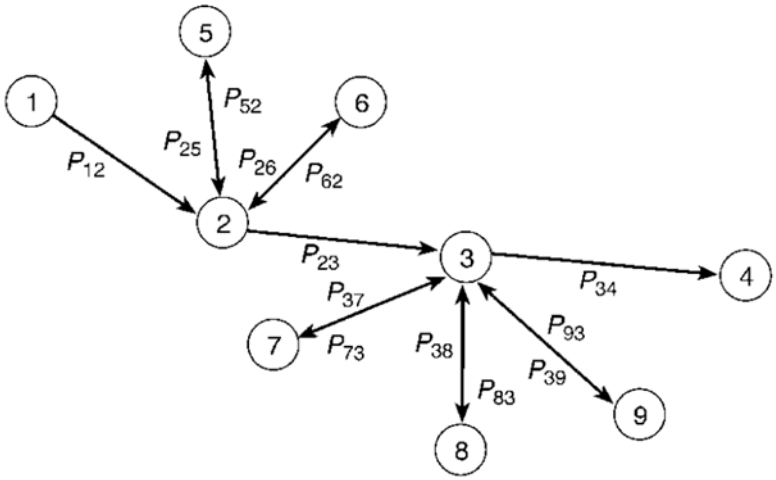


Figura 4 Flujo 'cargo and baggage handling'

Para el modelado de los tiempos se utilizan distribuciones de probabilidad, tales como Exponenciales, Beta y Normal. El estudio se basa en el análisis de datos históricos para determinar cuáles de ellas se han de utilizar para cada uno de los diferentes estados.

States	*State Description*	*States*	*State Description*	*IATA Delay Codes and Description*
1	*Arrival*			
2	*Goods unloading*	*5*	Cargo Processing	22, 23, 26 Late positioning & preparation
		6	Aircraft Ramp Handling	32, 33 Lack of loading staff, cabin load Lack of equipment, staff/operators
3	*Goods loading*	*7*	Cargo Processing	22, 23, 26 Late positioning & preparation
		8	Aircraft Ramp Handling	32, 33 Lack of loading staff, special load Lack of equipment, staff/operators
		9	Passenger & Baggage	11, 12, 18 Late check-in, check-in congestion Late baggage processing
4	*Departure*			

Tabla 2 Estados del flujo 'Cargo and baggage handling'

Además, se añaden al modelo cuatro eventos independientes causantes de retraso que suelen ser frecuentes en el proceso del *turnaround*: retraso en la carga de combustible, retraso en las comprobaciones de mantenimiento, daños en la aeronave y cambios de aeronave.

Este modelo se ofrece como una posible aplicación para las compañías aéreas para poder analizar su escala y conocer dónde se encuentran los puntos débiles.

Jürgen Kuster et al. (2008) propone otra forma para afrontar el problema: el *Extended Resource-Constrained Project Scheduling Problem* (*x-RCPSP*).

El x-RCPSP, es una mejora del *Resource Constrained Project Scheduling Problem* (*RCPSP*), un algoritmo muy utilizado para la resolución de problemas de planificación, muy centrado en el ámbito de la producción industrial.

El RCPSP no ofrece la oportunidad de reconsiderar las decisiones tomadas previamente sobre la ejecución de una actividad, por lo tanto no existe la posibilidad de realizar cambios sobre la planificación. Para la gestión de las perturbaciones resulta necesario poder realizar dichos cambios cuándo sea oportuno.

El x-RCPSP surge ante la necesidad de gestionar las perturbaciones inherentes en el *turnaround*, extendiendo el RCPSP para gestionar dicha materia. El concepto del x-RCPSP se basa en la definición de **actividades alternativas**.

Las actividades alternativas son formas de realizar la misma actividad de una manera más rápida o eficaz aplicando más recursos en ella, por ejemplo, la actividad alternativa de realizar el desembarque de pasajeros por una puerta sería realizarlo por dos puertas (aunque el coste sería más elevado, el tiempo de proceso se vería altamente reducido).

La Figura 5 muestra un ejemplo de proceso de *turnaround* en el que para las actividades que se consideran críticas (Desembarque, Carga de Combustible /Catering /Limpieza y Embarque) se definen sus actividades alternativas. Estas actividades alternativas pueden estar **activas** o **inactivas** en función de las necesidades concretas de cada momento y los nodos *xor* expresan la posibilidad de utilizar la ejecución de las actividades alternativas.

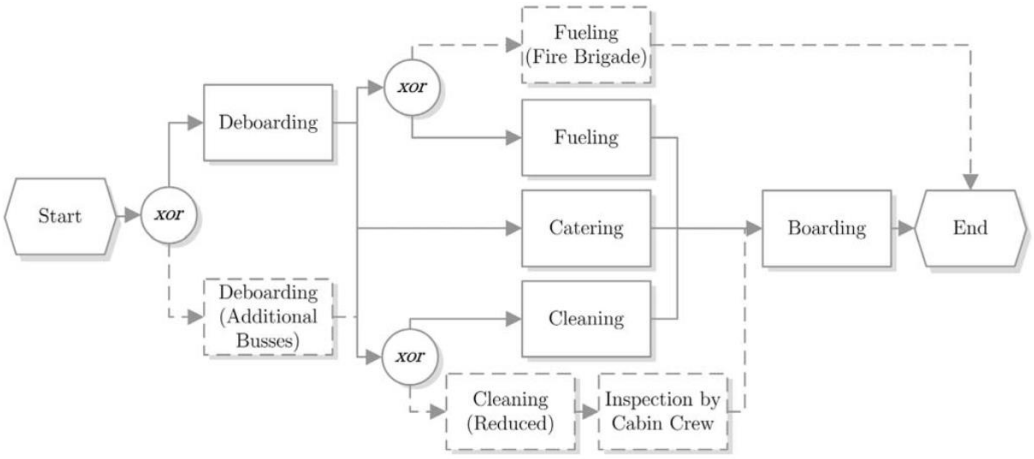


Figura 5 Ejemplo de proceso del turnaround con modificaciones posibles (Kuster,2008)

El x-RCPSP ofrece una adaptación dinámica durante el desarrollo real de las operaciones que permite minimizar el retraso, p.ej. si un avión llega con retraso a la puerta de embarque, se puede optar por estas actividades alternativas que permitirán reducir el efecto de la acumulación de los retrasos.

Makhloof & Waheed (2014) Presentan en su artículo *Optimization for Aircraft Turnaround Operations using Project Evaluation and Review Technique*, el **FAPS** (*Flight Activities Progressions systems*), una herramienta creada con el objetivo de gestionar y monitorizar la escala de aeronaves con el objetivo de minimizar su tiempo total.

El FAPS está basado en PERT (*Project Evaluation and Review Technique*) y en su utilización para la detección del camino crítico. Se basa en el concepto de que el camino crítico no es siempre el mismo.

Tal y cómo **Cheng-Lung Wu (2010)** define en su estudio de análisis de la escala mediante PERT, a cada actividad del proceso se le puede añadir un $Slack_i$ que define lo que se puede demorar esa actividad sin que afecte a la duración total del proceso. En muchos casos ese tiempo es realmente pequeño por lo que un mínimo imprevisto puede provocar que el camino crítico varíe.

Teniendo esto en cuenta, **Makhloof & Waheed (2014)** crean el FAPS, un sistema que va actualizándose con la información que se va registrando durante el proceso y permite al coordinador de la escala conocer el estado real del proceso. El FAPS envía mensajes al coordinador con el estado del proceso en función de la planificación, lo cual le permite realizar las tareas de gestión de una forma más eficaz.

Se define el FAPS cómo un sistema basado en reglas, con los módulos principales que se muestran en la Figura 6. El usuario se comunica con el sistema mediante la interfaz de usuario (*User Interface*) y entonces el *inference engine* resuelve el problema utilizando la base de conocimiento y los datos del problema en concreto. La base de conocimiento funciona mediante reglas IF-THEN, mientras que los datos específicos del caso contienen datos introducidos por el usuario y conclusiones parciales basadas en esos datos.

Existen además otros dos módulos, el módulo del sistema de explicación (*Explanation system*) y el editor de la base de conocimiento (*Knowledge base editor*). El módulo de *Explanation system* permite al programa explicar el razonamiento que ha seguido al usuario, mientras que el módulo de *Knowledge* base editor ayuda al ingeniero del sistema actualizar y revisar la base de conocimiento.

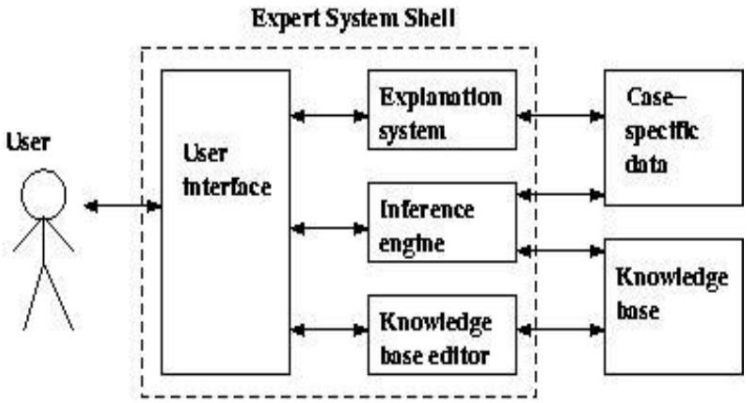


Figura 6 Arquitectura del FAPS

Para demostrar la funcionalidad del sistema FAPS, se realizó una implantación en la compañía EgyptAir desde Junio del 2003 hasta Marzo del 2011. Durante este período se realizan diferentes tipos de análisis sobre los datos observados y se concluye que la implantación del sistema FAPS resulta beneficiosa para las compañías aéreas y las compañías de *handling*. Algunos de los hitos más importantes que se consiguen aplicando este sistema son:

[1] Minimizar el tiempo que la aeronave esta parada en tierra (de 120 minutos a 60 minutos y a 45 minutos en caso de aviones de tamaño medio) y maximizar la utilización de la flota.

[2] Reducir el factor humano cómo factor de retraso.

[3] Ahorrar costes y minimizar las comunicaciones manuales.

[4] Evaluar el desarrollo mensual y anualmente para cada actividad.

[5] Buen desarrollo del sistema de alertas esperadas utilizando el método del CPM (Critical Path Method).

[6] Minimización del impacto de las demoras para restaurar el OTP (On-Time Performance).

[7] Exposición de las causas de retraso y su impacto sobre la planificación del resto del día.

[8] Proporciona la información necesaria para las decisiones de gestión correctas.

B. Oreschko et al. (2012) presentan el concepto ***Ground Manager*** **(GMAN)** en el ámbito del ***Airport Collaborative Decision Making*** **(A-CDM)** con el objetivo de minimizar el tiempo de escala utilizando las herramientas más desarrolladas del AMAN y DMAN (Figura 7).

Se definen tres factores esenciales para el desarrollo del DMAN:

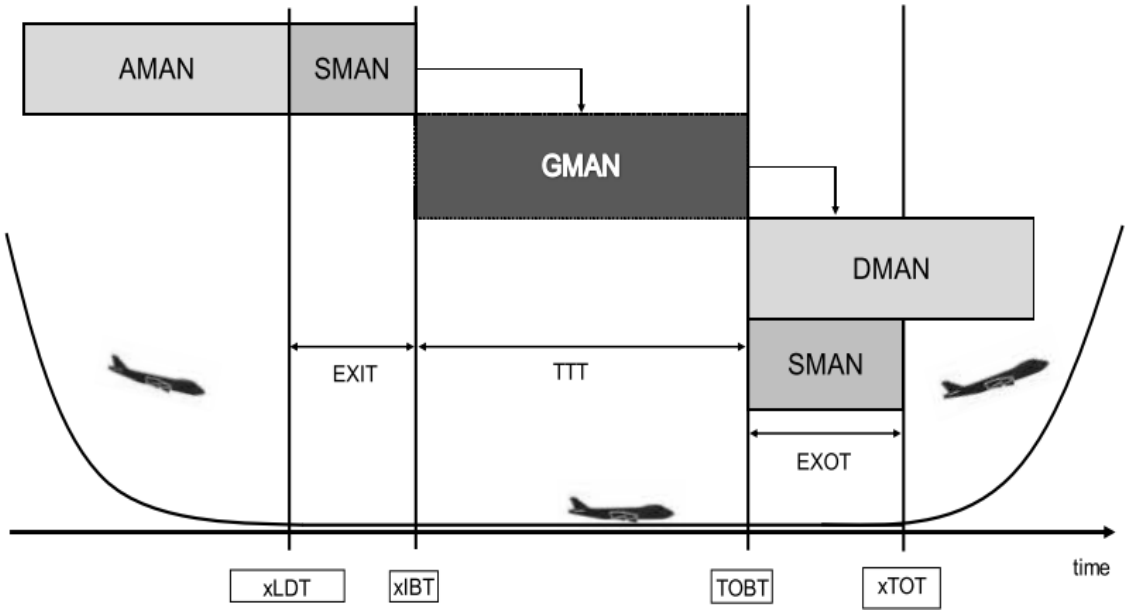


Figura 7 Concepto GMAN (Oreschko et al2012)

1. Las actividades que forman el *turnaround* se deben modelar para que su tiempo de proceso sea **estocástico**.
2. El proceso de *turnaround* **depende** de varios parámetros p.ej. tipo de aeropuerto, número de pasajeros, aerolínea, etc.
3. Los retrasos de llegada tienen una influencia importante en el tiempo de desarrollo de cada una de las actividades y de la interacción entre ellas.

El GMAN se basa en el concepto del camino crítico, es decir, aquella sucesión de actividades que precisa de más tiempo para completarse. El proceso crítico identificado en este artículo es el siguiente: Desembarque, Carga de Combustible /Catering /Limpieza y Embarque.

Éstas actividades tienen tiempos de desarrollo estocásticos, para ello las herramientas del DMAN utilizan diversas distribuciones estadísticas, aunque la más utilizada es la distribución *Weibull*.

Para garantizar la secuencialidad de las actividades, si el tiempo de finalización de una actividad es mayor al tiempo de inicio estimado de la siguiente, el tiempo de inicio de la siguiente tarea se desplaza al tiempo de finalización de su predecesora. De esta forma se asegura la integridad del modelo.

El tiempo total del *turnaround* se puede definir, entonces, cómo la suma de las actividades (Figura 8).

- TTT = (start-time (deboarding) + duration (deboarding))
  + (MAX process end-time of
  ((start-time (catering) + duration (catering))
  or (start-time ((fueling) + duration (fueling)))
  or (start-time ((cleaning) + duration (cleaning)))
  + start-time ((boarding)+duration (boarding))

Figura 8 Cálculo del tiempo total de turnaround, B. Oreschko et al. 2012

La aplicación del concepto GMAN es predecir el TOBT (*Target Off Block Time*) que es el tiempo en el que finalizará la escala de la aeronave, basándose en la estimación del IBT (*In Block Time*) o tiempo de llegada de la aeronave y de TTT (*Turnaround Time*) que es el tiempo que tarda en producirse la escala.

Se definen cuatro niveles de predicción basados en el tiempo que queda para el EIBT (*Estimated In Block Time*). A medida que nos acercamos al EIBT la cantidad de datos utilizables para la predicción son mayores.

Los cuatro niveles LAT (*Look Ahead Time*) son:

1. **Básico**: Más de 6 meses antes del EIBT.
2. **Estratégico**: Entre siete días y 6 meses previos al EIBT.
3. **Táctico**: Menos de siete días al EIBT.
4. **Actual**: Después del AIBT (Actual IBT).

Para demostrar la funcionalidad del modelo, **B. Oreschko et al. (2012)** lo implementan en una aplicación Java. El prototipo estudia 4 escenarios diferentes correspondientes a los cuatro niveles de predicción LAT para un vuelo concreto.

Los tipos de datos utilizados provienen de diferentes fuentes para cada uno de los niveles de predicción, tal y cómo se muestra en la Figura 9.

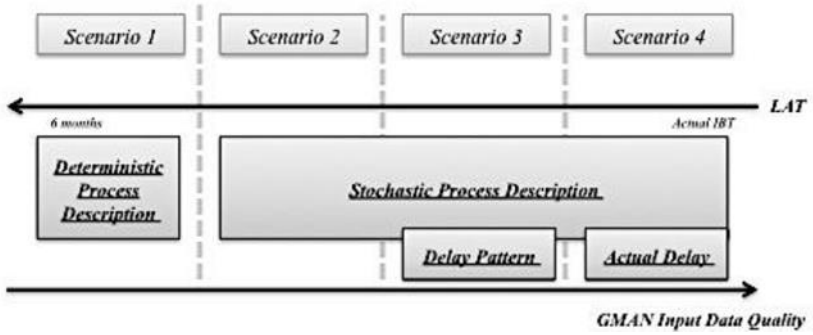


Figura 9 Inputs utilizados en los diferentes niveles de predicción. B. Oreschko et al. (2012)

Tras realizar el estudio se obtienen diferentes resultados para cada uno de los escenarios (Tabla 3). Tal y cómo se puede observar, a medida que la estimación se acerca más al EIBT la estimación de la duración del *turnaround* es más precisa debido a que se tienen en cuenta más variables (La estimación en la fase básica es de 38 minutos mientras que en la fase actual es de 23 minutos).

	TTT Duration and Times			
	Q .25	Mean	Q .50	Q.75
Scenario 1		38 min		
Scenario 2	31 min	32 min	33 min	35 min
Scenario 3	30 min	31 min	31 min	32 min
Scenario 4	22 min	23 min	23 min	25 min

Tabla 3 Comparación de escenarios, B. Oreschko et al. 2012

5.2 Segundo Problema: Añadir un buffer temporal

El método más utilizado en la actualidad por las compañías aéreas para optimizar el *turnaround* consiste en añadir un *buffer* de tiempo al final de todo el proceso. El problema de esta metodología reside en que no es posible definir un buffer estándar que se pueda utilizar para cualquier tipo vuelo, debido a que hay un grado de incertidumbre muy elevado en las operaciones de este tipo.

Resulta necesario entonces la búsqueda de un *buffer* dinámico que se adapte a las diferentes situaciones posibles y que a la vez permita optimizar al máximo el tiempo y el coste del *turnaround*.

Fricke & Schultz (2009) plantean una forma de afrontar este problema. Realizando una investigación sobre las prácticas actuales y proponen una estrategia para definir mejor los *buffers* temporales.

El concepto de **Fricke & Schultz** se basa, no únicamente definir un tiempo extra al final del *turnaround*, sino que a la vez, definir ciertos *buffers* en las actividades que componen el camino crítico. A primera vista esto parece alargar el tiempo de desarrollo final, pero tal y cómo se concluye en este estudio esto no resulta ser así.

El estudio se basa en desarrollar una metodología en la que no únicamente se añada un buffer al final del *turnaround*, sino que teniendo en cuenta los tiempos de interferencia entre actividades que se acumulan a lo largo de la escala se puede optimizar el *buffer* total. Debido básicamente a que los tiempos de interferencia pueden absorber buena parte del tiempo total que únicamente absorbía el *buffer* total.

Para el desarrollo del algoritmo de resolución, se modelan los tiempos de cada actividad del camino críticos con tres funciones estadísticas (Weibull, Gamma y Normal).

Delay	Min. TA (min)	Total Buffer (min)	TA with Buffer (min)	Expected TA (min)	Process Inter-actions (min)
Scheduled	38	17.78	55.78	57.93	2.15
5 – 10 minutes	38	12.85	50.85	54.03	3.18
10 – 15 minutes	38	9.64	47.64	50.97	3.33
15 – 20 minutes	38	7.27	45.27	49.31	4.04
20 – 25 minutes	38	9.62	47.62	51.03	3.41
25 – 30 minutes	38	5.47	43.47	48.83	5.36

Tabla 4 Tiempos de turnaround en función del retraso de llegada, Fricke & Schultz (2009)

En la Tabla 4 se exponen los resultados del estudio realizados mediante simulación MC. Se observa en ésta tabla que en cuanto mayor es el tiempo de retraso de llegada, el tiempo esperado de *turnaround* es menor. Aunque el tiempo de interacción entre procesos aumenta, el tiempo total del *turnaround* no lo hace. Esto se debe a que podemos disminuir el tiempo de buffer al final. Por ejemplo, en el caso de retraso de 25-30 minutos, debido a que hay un retraso acumulado muy elevado, los actores implicados emplean menos tiempo para realizar las actividades debido a la presión de querer cumplir con el tiempo planificado.

La conclusión que se puede extraer de este estudio, es que a medida que el vuelo llega con más retraso, menos *buffer* total es necesario añadir para que se cumpla la planificación.

Wu & Caves (2004) estudian la forma de planificar un *turnaround* con el objetivo de minimizar los costes de retraso para los pasajeros y las compañías aéreas.

Este estudio define todos los costes asociados al retraso mediante fórmulas matemáticas, cómo por ejemplo, costes de oportunidad, costes asociados al retraso del vuelo anterior, costes de los pasajeros, etc. Una vez definidos, utiliza distribuciones para estimar cuáles serán sus valores ya que no se trata de costes fijos, sino de costes que variarán con el tiempo.

El objetivo final del estudio es el de conocer cómo se comportan los costes, lo cual permitirá planificar de una forma correcta el tiempo que ha de tener el *buffer*.

El modelo de **Wu & Caves** se basa en el esquema de la Figura 10, en el que se representa el tiempo planificado de buffer (*T*) y el retraso de llegada (*t*).

C.-L. WU AND R. E. CAVES

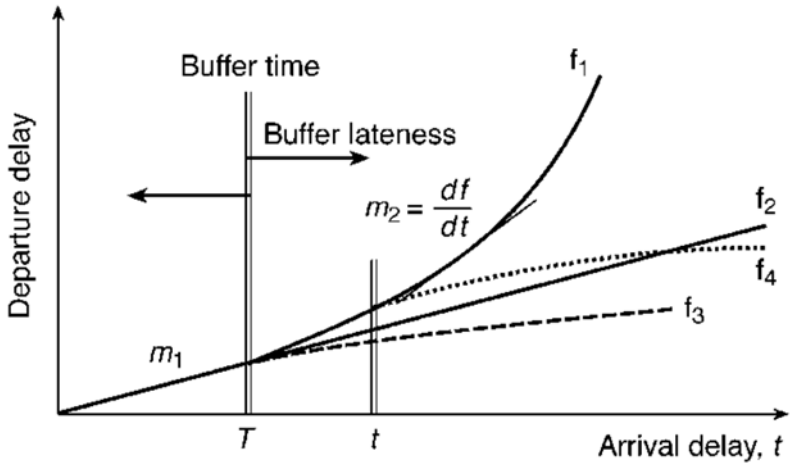


Figura 10 Desarrollo de los retrasos de salida, Wu & Caves (2004)

Si t < T entonces no surge ningún problema, ya que el tiempo de buffer planificado será capaz de absorber el retraso. En cambio, si t > T se pueden dar tres situaciones diferentes:

- Curva f_1: El número de salidas con retraso se ve aumentado debido a una mala gestión de las llegadas con retraso. La curva f4 representa la función más optimista ante esta situación.
- Curva f_2: El número de llegadas con retraso es linealmente proporcional con el de salidas con retraso.
- Curva f_3: el número de salidas con retraso disminuye, lo cual se puede atribuir a una buena gestión de *handling* que permite disminuir el efecto creado por las llegadas con retraso.

La conclusión de este estudio es que se ha de buscar un punto de equilibrio, en función de las diferentes características de cada aeropuerto y de cada aerolínea, a la hora de definir un buffer temporal. Si se define un buffer demasiado ajustado, se corre el riesgo de incurrir los costes generados por los retrasos (p.ej. cargos por uso de puerta de embarque o costes de personal). Si se define un buffer temporal con demasiada holgura existe el riesgo de incurrir en los costes de oportunidad, es decir, lo que la aeronave deja de ganar por estar parada en tierra esperando a su salida.

Sección 3: Definición del problema y Descripción del algoritmo

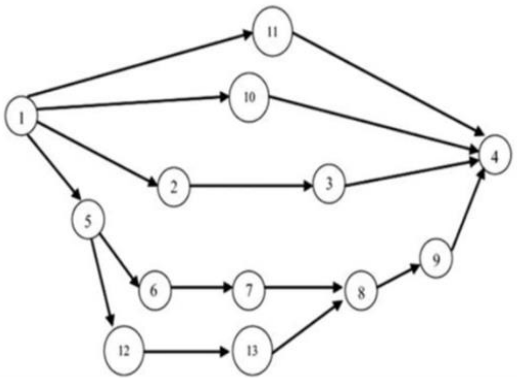
6 DEFINICIÓN DEL PROBLEMA

El problema consiste en detectar los cuellos de botella en los flujos de trabajo del proceso de escala, es decir, aquellos procesos que duran más tiempo y por tanto el proceso no finaliza hasta que estos procesos hayan finalizado.

Para esto se ha desarrollado un algoritmo capaz de analizar el proceso y la duración de cada actividad, ofreciendo un análisis sobre una escala de aeronave concreta.

Para desarrollar el algoritmo, en primer lugar se ha utilizado en un algoritmo ya existente, el SREMS. Un algoritmo utilizado para calcular la fiabilidad en sistemas complejos desarrollado por **Angel A. Juan et. al (2004)**. Este algoritmo presenta unas características que se asemejan a las buscadas para el desarrollo del algoritmo propio. Se han modificado ciertas características del algoritmo para adaptarlas a las necesidades de este proyecto.

En segundo lugar, para definir los inputs que serán necesarios para el algoritmo, este estudio se ha basado en la representación que hace **Cheng-Lung Wu (2010)** al representar la escala de aeronaves en forma de red. Se ha utilizado dicha estructura para definir los caminos que existen en todo el proceso, así como las actividades que forman parte del mismo. En la Figura 11 se puede observar la representación en forma de red.



Node	Activity	Duration
1	Aircraft on stand (start)	0
2	Cargo/baggage off-loading	20
3	Cargo/baggage loading	25
4	Push back/taxi (finish)	0
5	Passengers disembarking	10
6	Cabin cleaning	15
7	Security procedures	3
8	Passenger boarding	15
9	Crew checklist procedures	3
10	Aircraft fuelling	25
11	Maintenance check	25
12	Catering off-loading	10
13	Catering loading	10

Figura 11 Representación de la escala en forma de red, Wu (2010)

El algoritmo que se explica a continuación se ha desarrollado en el lenguaje de programación C++ con la herramienta Microsoft Visual Studio 2010.

7 FUNCIONAMIENTO DEL ALGORITMO *T-CPA*

El algoritmo se ha denominado **T_CPA** debido a las siglas de ***Turnaround Critical Path Analysis***. A continuación se define el funcionamiento del mismo.

Inputs

La introducción de los inputs al programa se realiza mediante la lectura de tres archivos de texto, los cuáles ha de rellenar el usuario con los datos explicados a continuación.

En el primer fichero de texto, se introducen los siguientes datos en el orden descrito: número de iteraciones que ha de realizar la simulación, número de componentes o actividades del sistema, número de instantes objetivo, distancia entre los instantes objetivo, número de caminos del sistema y la semilla (para la simulación de números aleatorios). Si se introduce el valor 0 cómo semilla, el algoritmo produce un número aleatorio basado en el reloj del ordenador.

En el segundo fichero, se ha determinar para cada una de las actividades la distribución de tiempo que mejor represente dicha actividad. Una vez introducida la distribución, también se ha de introducir la longitud que tiene cada camino del sistema.

En el tercer fichero se ha de añadir únicamente la composición que tiene cada uno de los caminos del sistema.

El orden de los ficheros y la información ha de ser definida de esta forma, sino se hace así el algoritmo no podrá redimensionar los outputs y la simulación no se realizará de forma correcta.

Introducción de los datos al algoritmo

La primera parte del T-CPA consiste en ir abriendo los ficheros previamente definidos. Se puede observar que, por ejemplo, en el caso de los datos del segundo fichero, para incorporarlos en el sistema es necesaria alguna información (la que se encuentra en el primer fichero).

Es por eso que cada vez que se abre un fichero y se cogen los datos, el algoritmo redimensiona sus variables para que sea capaz de adquirir los datos del siguiente fichero.

Dimensionar el problema

Una vez ya se han introducido todos los datos, el algoritmo redimensiona las estructuras que se utilizarán para la simulación y la generación de los outputs.

Simulación

La simulación realiza para cada iteración los siguientes cálculos:

En primer lugar genera un tiempo aleatorio para cada una de las actividades. Esta operación se realiza llamando a una función en la que están definidas diferentes funciones de distribución. El algoritmo detecta la distribución que debe aplicar (información añadida en el segundo fichero) y para cada actividad, la cual tiene asociada unos parámetros, genera un tiempo aleatorio.

Una vez calculados los tiempos aleatorios de las actividades, para cada uno de los caminos calcula su tiempo total sumando los tiempos de las actividades que lo componen. Una vez ha calculado el tiempo de un camino, lo compara con las observaciones de los otros caminos de esa iteración, y almacena el número del camino que ha sido más largo y el del camino que ha sido más corto. Esta operación se realiza tantas veces cómo iteraciones se hayan definido. Al final de la simulación obtenemos para cada una de las iteraciones, cuál ha sido el camino más largo y cuál ha sido el más corto.

Cálculo de los outputs

Una vez ya tenemos los resultados de la simulación, el algoritmo realiza una serie de cálculos sobre los datos utilizando fórmulas estadísticas. Los cálculos más importantes que realiza son el cálculo de la media de tiempo de finalización del proceso, la desviación típica estimada y el intervalo de confianza al 99%.

Además calcula la probabilidad de que el sistema haya finalizado en los instantes objetivos definidos en el primer fichero de inputs.

Resumen para usuario

La última parte del algoritmo consiste en abrir (o crear) un fichero de texto en el que se imprimen los resultados de la simulación. El formato del fichero de outputs datos es el que se muestra en la Tabla 5.

```
SIMULATION OUTPUTS:
2 48.810200 2.585634 48.780414 48.839986

1:    100.000000 100.000000 100.000000
2:    100.000000 100.000000 100.000000
3:    100.000000 100.000000 100.000000
4:    100.000000 100.000000 100.000000
5:    100.000000 100.000000 100.000000
6:    100.000000 100.000000 100.000000
7:    100.000000 100.000000 100.000000
8:    99.984000 99.969430 99.998570
9:    93.980000 93.706002 94.253998
10:    31.174000 30.640414 31.707586

Path     Largest Shortest   MIN    MAX    MEAN
1:       0.00    97.95    10.945 24.284  19.996
2:       0.00    2.05     13.997 29.132  25.001
3:       18.36   0.00     30.747 52.375  45.002
4:       17.16   0.00     29.946 56.189  45.986
5:       64.48   0.00     29.941 60.781  48.006

Activity  MIN        MAX       MEAN
1:       | 11.106 | 24.616 | 20.000
2:       | 14.389 | 29.517 | 25.003
3:       | 1.969 | 15.090 | 9.995
4:       | 6.437 | 19.755 | 14.994
5:       | 1.086 | 3.937 | 2.999
6:       | 5.137 | 19.507 | 14.998
7:       | 1.302 | 3.877 | 3.000
8:       | 13.997 | 29.132 | 25.001
9:       | 10.945 | 24.284 | 19.996
10:      | 1.946 | 15.127 | 10.002
11:      | 2.607 | 14.951 | 10.010
```

Tabla 5 Ejemplo de Outputs del T-CPA

La tabla nos indica lo siguiente:

- La primera fila indica, respectivamente, el tiempo de simulación, el tiempo medio de finalización del proceso, la desviación típica estimada y los intervalos de confianza al 99%.
- Debajo se muestra el cálculo de probabilidad de que el sistema siga funcionando en los instantes objetivos. Por ejemplo, en el instante objetivo 8 (40 minutos) la probabilidad de que el sistema siga funcionando es de un 99.98% mientras que en el instante objetivo 10 (50 minutos) es de un 34.18%.
- El siguiente grupo de datos son los resultados de la simulación en lo que se refiere a los procesos o caminos (*Path*). Para cada uno de estos caminos se muestra el porcentaje en el que éste ha sido el más largo (primera columna) y en el que ha sido el más corto (segunda columna). Además también muestra aquellos tiempos mínimos y máximos de todas las iteraciones para cada camino. Por último la media de duración de cada camino.
- El último agregado de datos son los resultados de la simulación para cada una de las actividades. Para cada una de las tareas se muestran los tiempos mínimos y máximos que han alcanzado en la simulación, además del tiempo medio de duración de cada actividad.

7.1 Cálculo de los parámetros de la distribución *Weibull.*

Una de las principales dificultades de este proyecto se encuentra en conseguir los parámetros que se deberán utilizar en la simulación para representar los tiempos. Estos parámetros se deben ajustar lo máximo posible a la realidad para que los resultados obtenidos del T-CPA resulten útiles y sean veraces.

La práctica normal para trabajar con este tipo de problemas suele ser la de obtener los tiempos medios y los resultados a través de unos parámetros ya conocidos, al contrario del problema aquí presentado.

Para la obtención de estos parámetros, se ha desarrollado una herramienta capaz de obtener los parámetros de una distribución *Weibull* mediante una media y varianza conocida. La solución consiste en una hoja *excel* (Ilustración 5) en la que se define el problema cómo un problema de optimización, resuelto mediante el *solver* de *excel*.

Compute Weibull parameters from a given mean and variance						
Lambda (scale):		10,6856	<- value to change			
k (shape):		7,0583	<- value to change			
		Obtained	**Desired**	**Diff in ABS**		
	Mean =	10,00	10	0,00		
	Var =	2,78	2,78	0,00		
	StDev =	1,67		**0,00**	<- value to minimize	

Ilustración 5 Herramienta de obtención de los parámetros de la distribución Weibull.

En la columna *Obtained* están definidas las ecuaciones de obtención de la media y la varianza, las cuáles se obtienen mediante los parámetros (*Scale* y *Shape*). Cómo se ha comentado anteriormente, éstos valores son los que queremos obtener por tanto no sabemos cuáles son.

$$Media: \lambda\Gamma\left(1+\frac{1}{k}\right)$$

$$Varianza: \lambda^2\Gamma\left(1+\frac{2}{k}\right)-\mu^2$$

Ecuación 1 Media y varianza de la Weibull

Lo que se ha de hacer es rellenar las columnas de *Scale* and *Shape* con cualquier valor positivo y anotar en la columna *Desired* los valores que se conocen (media y varianza). En la columna *Diff in ABS* se calcula la diferencia entre la media y varianza que se obtiene con los parámetros y la que se desea que sea. El valor a minimizar es la suma de ambas diferencias.

El problema de optimización es el siguiente:

$$\begin{aligned} &Minimize\ \boldsymbol{Diff\ in\ ABS};\\ &Subject\ to: Lambda \geq 0;\\ &K \geq 0; \end{aligned}$$

El *solver* va cambiando los valores de Lambda y K, con la restricción de ser siempre positivos, hasta que éstos dan una media y una varianza igual o muy similar a la deseada. Es decir, que la diferencia entre ambas es mínima.

El proceso normal para obtener estos parámetros en una compañía aérea o empresa de *handling* consistiría en adquirir los tiempos de duración de las actividades de la escala e ir almacenándolas durante un cierto período de tiempo. Tras finalizar ese período, se calcularía la media y la varianza de los datos y se podrían obtener los parámetros utilizando una herramienta cómo la aquí presentada.

Sección 4: Experimentos computacionales

8 REALIZACIÓN DE EXPERIMENTOS

En este apartado se realizan una serie de experimentos con el algoritmo T-CPA de detección de los cuellos de botella para la escala de diferentes tipos de aeronaves.

En el presente trabajo se han utilizado las escalas propuestas por los fabricantes para las siguientes aeronaves:

- Boeing 737-800
- Airbus 320-200
- Airbus 380-800
- Boeing 747-8F
- Boeing 757-200PF

A partir de las escalas propuestas por los fabricantes en los manuales de la aeronave se han realizado las adaptaciones para poder ser analizadas gracias al algoritmo, manteniendo siempre la secuencialidad de las actividades.

En el caso del **Boeing 737-800** y el **Airbus 380-800** se han analizado, además, las escalas teniendo en cuenta diferentes **ratios de ocupación** (75%, 90% y 100%), para analizar la forma en que varían los tiempos y los cuellos de botella en estas escalas.

El mayor problema a la hora de realizar las pruebas es el de encontrar la información referente a las **varianzas de los tiempos** de las actividades de cada aeronave. Esta información es muy variable para los usuarios de la aeronave, por lo cual resulta lógica la dificultad de obtener este tipo de datos.

Para poder realizar los experimentos en este trabajo se han utilizado las varianzas que utiliza **Cheng-Lung Wu (2010)** en su estudio, ya que las actividades básicas son las mismas.

8.1 Aeronaves de Transporte de Pasajeros

8.1.1 Boeing 737-800

La serie del Boeing 737 es la serie de aviones más vendida de la historia de la aviación. Aeronave bimotor utilizada para vuelos de medio-largo alcance y popularizada, sobre todo, por su utilización por las compañías de *low-cost*.

El Boeing 737-800 pertenece a la serie *Next Generation*, todavía en producción en la actualidad. Además es la serie utilizada por la conocida aerolínea *low-cost* Ryanair.

La compañía Boeing propone la escala que se muestra en la Ilustración 6 para el Boeing 737-800.

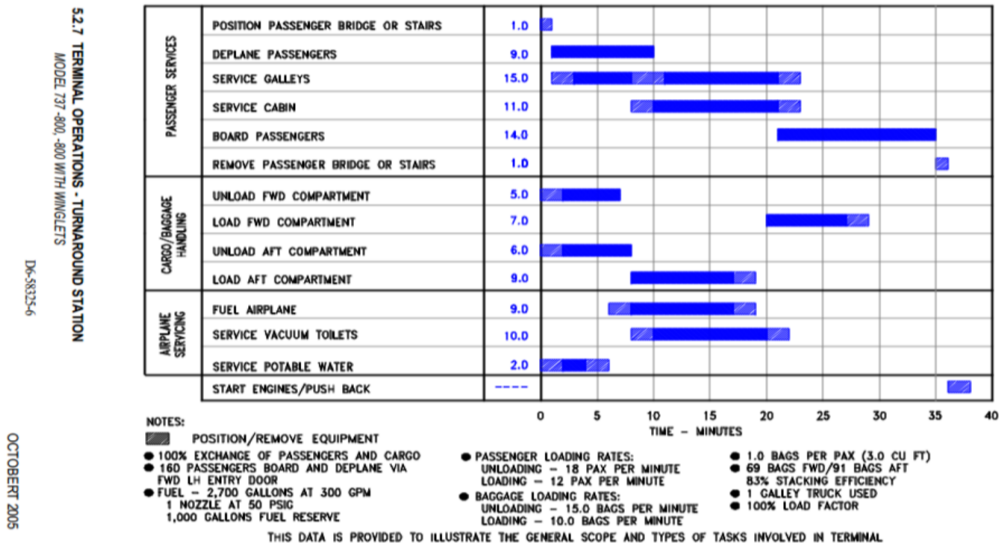


Ilustración 6 Escala propuesta por Boeing para el B737-800 (Boeing,2005)

En la escala propuesta por Boeing se pueden observar, a priori, 3 flujos de trabajo: Por un lado las actividades que desarrollan en cabina, las actividades de *handling* y las actividades de servicio técnico a la aeronave.

Si se representa la escala en forma de red, tendría una estructura cómo la que se puede observar en la Figura 12, en la que hay 5 caminos paralelos entre sí. Los caminos son los siguientes:

[1] Desembarque de pasajeros-Limpieza-Embarque de pasajeros.

[2] Catering.

[3] Descarga de carga y equipaje – Carga de carga y equipaje.

[4] Descarga de carga y equipaje – Combustible.

[5] Descarga de carga y equipaje – Mantenimiento.

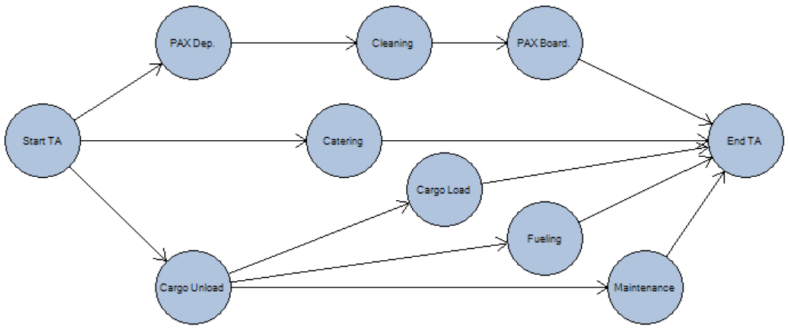


Figura 12 Representación de la escala del B737-800 en forma de red

B737-800			Oc 100%	Oc 90%	Oc 75%
	Number	Activities	minutes	minutes	minutes
Passenger Services	1	Deplane passenger	9	8	7
	2	board passenger	14	12	10
	3	Catering (Service Galleys)	15	15	15
	4	Cleaning (Service Cabin)	11	11	11
Cargo handling	5	Unload cargo/baggage	6	6	6
	6	Load cargo/baggage	16	16	16
Airplane service	7	Fuel	9	9	9
	8	Maintenance	10	10	10

Tabla 6 Tiempos medios de las actividades del B737-800 con diferentes ratios de ocupación

En la Tabla 6 se muestran los tiempos medios estimados de las actividades de la escala del B737-800. También se muestran los tiempos teniendo en cuenta 3 ratios de ocupación de pasajeros. El cálculo de los tiempos se ha realizado utilizando las fórmulas que se detallan en el manual de la aeronave.

Una vez utilizado el algoritmo T-CPA con estos 3 escenarios se obtienen los siguientes resultados:

Ocupación 100%		
Mean TA: 33,92 min.		
Paths	Largest	Shortest
path1	99,91%	0,00%
path2	0,00%	41,70%
path3	0,09%	0,06%
path4	0,00%	40,12%
path5	0,00%	18,12%

Ocupación 90%		
Mean TA: 31 min		
Paths	Largest	Shortest
path1	99,08%	0,00%
path2	0,00%	41,82%
path3	0,92%	0,08%
path4	0,00%	40,03%
path5	0,00%	18,07%

Ocupación 75%		
Mean TA: 28,1 min		
Paths	Largest	Shortest
path1	94,66%	0,00%
path2	0,00%	41,65%
path3	5,34%	0,05%
path4	0,00%	40,38%
path5	0,00%	17,92%

Tabla 7 Outputs del B737-800 en tres escenarios diferentes

Se puede observar que el tiempo total de la escala se reduce a medida que disminuye la ocupación, por lo que se puede decir que los procesos que están directamente relacionados con los pasajeros son los críticos. En este caso, el primer camino (Desembarque de pasajeros-Limpieza-Embarque de pasajeros) es el más largo de todos y por tanto es el camino crítico.

El estudio de los tres escenarios permite afirmar que el hecho de reducir la ocupación de la aeronave no implica que el camino crítico varíe: con un 75% de ocupación, el camino numero 1 sigue siendo el crítico en el 95% de las veces, el 5% restante de las veces es el camino 3 (Descarga y Carga de equipajes).

En los que se refiere al camino más corto vemos que no hay ninguna variación en función de la ocupación.

8.1.2 Airbus A380-800

El Airbus A380 es la aeronave comercial de transporte de pasajeros más grande del mundo en la actualidad. La aeronave consta de dos cubiertas a dos niveles a lo largo de su fuselaje.

La capacidad de pasajeros del A380 es muy variable en función de la serie y de la configuración de asientos que la aerolínea decida. Para realizar este estudio se han tomado los datos que se definen en el manual de la aeronave, un total de 555 pasajeros, los cuáles se distribuyen en 356 pasajeros en la cubierta inferior y 199 en la superior.

A la hora de analizar los tiempos de embarque y desembarque, se ha tomado en cuenta únicamente el tiempo de la cubierta inferior, debido a que los tiempos de embarque y desembarque en la cubierta inferior son siempre superiores a los del nivel superior, según las fórmulas definidas en el manual de la aeronave. Los embarques y desembarques de ambos niveles comienzan paralelamente, por lo que no resulta tener en cuenta los del nivel superior.

Esta misma operación se ha realizado también con las operaciones de catering.

Analizando la escala propuesta por Airbus se puede definir la escala cómo una red (Figura 13) con los siguientes caminos:

[1] Desembarque de pasajeros-Catering-Embarque de pasajeros.

[2] Desembarque de pasajeros-Limpieza-Embarque de pasajeros.

[3] Desembarque de pasajeros-Combustible.

[4] Mantenimiento.

[5] Descarga de equipaje y carga- Carga de equipaje y carga.

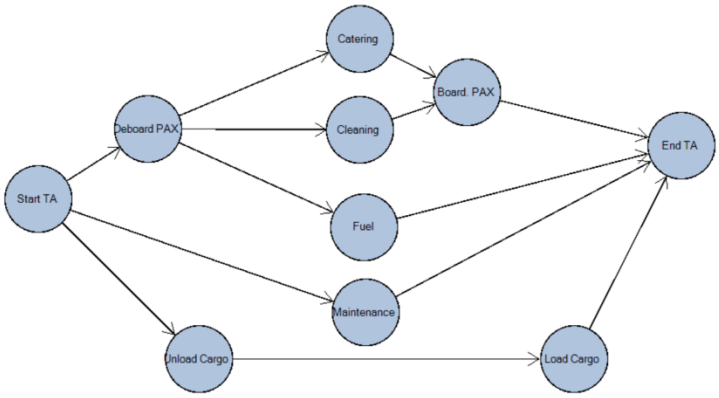


Figura 13 Escala representada en forma de red del A380-800

Para el análisis de la escala de A380-800 también se han definido 3 escenarios con diferentes ocupaciones (100%,90% y 75%). Los tiempos de las actividades se describen en la Tabla 8.

A380-800			Ocupación 100%	Ocupación 90%	Ocupación 75%
	Number	Activities	minutes	minutes	minutes
Passenger Services	1	Deboard passenger	16	13	11
	2	board passenger	25	22	18
	3	Catering (All)	45	45	45
	4	Cleaning (Service Cabin)	30	30	30
Cargo handling	5	Unload cargo/baggage	25	25	25
	6	Load cargo/baggage	30	30	30
Airplane service	7	Fuel	45	45	45
	8	Maintenance	60	60	60

Tabla 8 Tiempos de las actividades del A380-800 con diferentes niveles de ocupación

Una vez utilizado el algoritmo T-CPA con los 3 escenarios obtenemos los outputs (Tabla 9). Se puede observar que el camino 1 (Desembarque-Catering Embarque) es el camino crítico en los tres escenarios analizados. Por eso, una disminución de los tiempos de estas actividades, provoca directamente una disminución del tiempo de escala global.

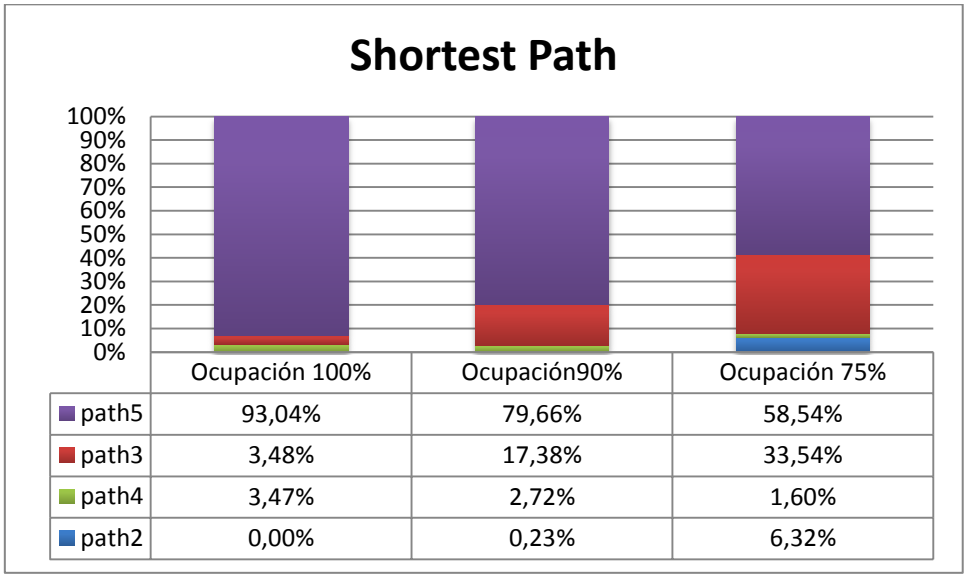
Ocupación 100%		
Mean TA: 86 min.		
Paths	Largest	Shortest
path1	100,00%	0,00%
path2	0,00%	0,00%
path3	0,00%	3,48%
path4	0,00%	3,47%
path5	0,00%	93,04%

Ocupación 90%		
Mean TA: 80 min		
Paths	Largest	Shortest
path1	100,00%	0,00%
path2	0,00%	0,23%
path3	0,00%	17,38%
path4	0,00%	2,72%
path5	0,00%	79,66%

Ocupación75%		
Mean TA: 74 min		
Paths	Largest	Shortest
path1	99,98%	0,00%
path2	0,00%	6,32%
path3	0,00%	33,54%
path4	0,02%	1,60%
path5	0,00%	58,54%

Tabla 9 Outputs del T-CPA de los 3 escenarios para el A380-800

En cambio sí que se puede observar una variación de los tiempos de los caminos más cortos. Tal y cómo se muestra en la Figura 14, cuando existe una ocupación máxima de pasajeros, el camino 5(Descarga y Carga de Equipajes y Carga) es el más corto en el 93% de los casos. A medida que la ocupación va disminuyendo, el camino número 3 (Desembarque de pasajeros y carga de combustible) pasa a ser en un 33.5% de los casos el camino más corto y el camino 5 pasa del 93% (con 100% de ocupación) al 58.5% (con un 75% de ocupación).



	Ocupación 100%	Ocupación90%	Ocupación 75%
path5	93,04%	79,66%	58,54%
path3	3,48%	17,38%	33,54%
path4	3,47%	2,72%	1,60%
path2	0,00%	0,23%	6,32%

Figura 14 Variación del camino más corto en función de la ocupación del A380-800

El análisis del camino más corto puede ser muy interesante para operadores de *handling* y aerolíneas, ya que en muchos casos reducir el tiempo del camino más largo puede resultar imposible. En estos casos, conocer cómo se comporta el camino más corto permite conocer que caminos terminarán antes y por tanto que recursos quedaran libres antes. Es decir, puede resultar interesante para optimizar recursos y por tanto reducir costes.

8.1.3 Airbus A320-200

El A320-200 es una aeronave de medio-largo alcance del fabricante Airbus. El A320-200 se puede considerar el competidor directo del B737 analizado anteriormente en este trabajo.

El hecho de ser el competidor directo del Boeing 737, hace que resulte interesante su estudio para conocer si existen ciertas similitudes en la escala de ambas aeronaves.

Tal y cómo cabe esperar, la estructura de la escala del A320-200 es similar a la del B737-800. En base a la escala propuesta en el manual de la aeronave, se puede definir la siguiente estructura (Figura 15).

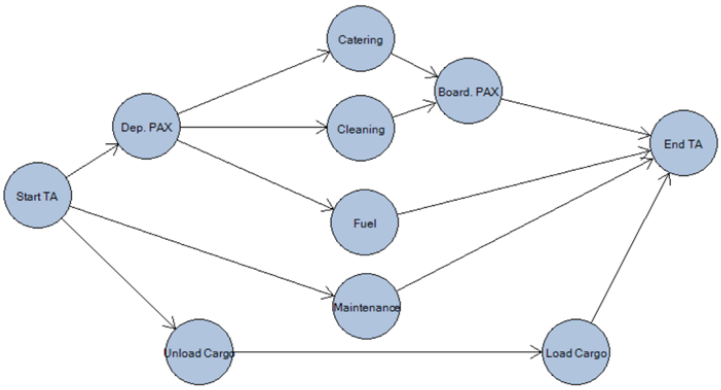


Figura 15 Estructura en forma de red de la escala del A320-200

Se pueden definir los siguientes caminos:

[1] Desembarque de pasajeros-Catering-Embarque de pasajeros.

[2] Desembarque de pasajeros-Limpieza-Embarque de pasajeros.

[3] Desembarque de pasajeros-Combustible.

[4] Mantenimiento.

[5] Descarga de Equipaje y carga- Carga de equipaje y carga.

En el caso de este estudio, debido a que únicamente se trata de conocer si existen diferencias entre la escala del A320 y el B737, no se ha considerado necesario hacer un análisis de ocupación cómo el que se ha realizado en los casos anteriores. Únicamente se ha analizado con un 100% de ocupación.

A320-200		
Mean TA: 35,04 min.		
Paths	Largest	Shortest
path1	94,57%	0,00%
path2	4,59%	0,00%
path3	0,84%	0,00%
path4	0,00%	75,61%
path5	0,00%	24,39%

Tabla 10 Outputs del T-CPA para el A320-200.

Tras definir los inputs con los tiempos de las actividades, se obtienen los outputs del T-CPA que se muestran en la Tabla 10. Se puede observar que el camino más largo en el 95% de los casos es el camino número 1 (Desembarque de pasajeros-Catering-Embarque de pasajeros) muy similar a los resultados del B737 (99%) con la única diferencia de que en el caso del Boeing en vez de la actividad *Catering* encontramos la actividad *Cleaning*.

Hay que tener en cuenta a la hora de buscar éstas diferencias entre los dos modelos que en el caso del A320, el manual de la aeronave define que las tareas *Catering* y *Cleaning* deben comenzar una vez ha finalizado el desembarque de los pasajeros, mientras que en el caso del B737 no define ese requisito para la actividad *Catering*.

En lo que se refiere al camino más corto, en el caso del A320, se encuentra sobretodo en el camino 4 (Mantenimiento) con un 75% de los casos. En cambio el camino más corto se encuentra más repartido en el caso del B737 (Tabla 7).

8.2 Aeronaves Cargueras

Para el estudio de aeronaves cargueras, se han seleccionado dos tipos de aeronave muy comunes en el ámbito de la carga aérea: El B747-8F y el B757-200PF.

8.2.1 Boeing 747-8F

El Boeing 747-8F nace de la adaptación de la aeronave de pasajeros B747 para el transporte de grandes cantidades de carga. Se utiliza sobretodo en el transporte de mercancías a grandes distancias debido al gran alcance que tiene esta aeronave.

Esta aeronave es la más utilizada por las compañías especializadas en carga aérea y que recorren distancias intercontinentales. Ésta gran utilización se debe a que permite diferentes tipos de carga (pallets y a granel).

Para realizar el estudio de esta aeronave se ha supuesto que transporta el máximo de su capacidad de carga en la bodega principal (34 pallets de 96'x125') y el máximo *bulk cargo*. Los tiempos medios estimados han sido extraídos del manual de la aeronave (Tabla 11).

B747-8F			
	Number	Activities	minutes
Cargo	1	Unload Main Deck	46
	2	Load Main Deck	46
	3	Unload FWD/AFT	20
	4	Load FWD/AFT	20
	5	Bulk Cargo	70
Aircraft Services	6	Fuel	45
	7	Manteinance	30

Tabla 11 Tiempos de escala del B747-8F

En la Figura 16 se puede observar la estructura en forma de red de la escala de la aeronave en la que se definen los 5 caminos siguientes:

[1] Descarga/Carga de la bodega principal.

[2] Descarga/Carga de los compartimentos delanteros y traseros.

[3] Carga a granel (*Bulk Cargo*).

[4] Combustible.

[5] Mantenimiento.

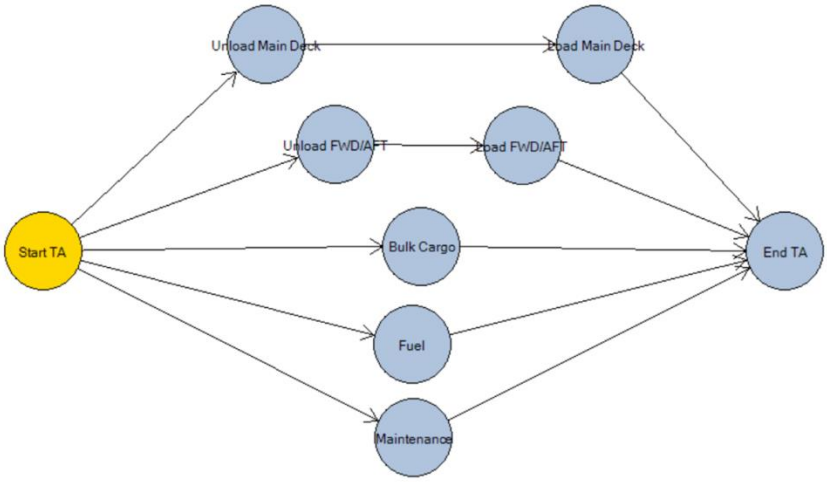


Figura 16 estructura en forma de red de la escala del B747-8F

Debido al gran tamaño de la bodega principal en la que se cargan los pallets, el tiempo de este camino siempre va a ser mucho más elevado que el resto. Es por eso que resulta más interesante conocer cuáles son los caminos más cortos para detectar aquellos recursos que quedan libres muy pronto y de esta forma utilizarlos para otras tareas. Es decir, optimizar los costes de la escala asociados a los equipos de trabajo ya que reducirlo en términos de tiempo puede resultar imposible o innecesario.

B747-8F		
Mean TA: 92 min.		
Paths	Largest	Shortest
path1	100,00%	0,00%
path2	0,00%	0,12%
path3	0,00%	0,00%
path4	0,00%	0,00%
path5	0,00%	99,88%

Tabla 12 Outputs del T-CPA para el B747-8F

Observamos, tal y cómo se esperaba, que el camino 1 es el más largo en la totalidad de los casos. Resulta muy interesante conocer, en cambio, cómo se comportan los caminos más cortos. Por ejemplo en este caso quizá sea posible que las actividades del camino 5 se realicen después de las del camino 4. De esta forma reducimos el número de caminos y el equipo restante (que antes se encargaba del camino 5) se puede utilizar cómo soporte para las actividades del camino 1.

8.2.2 Boeing 757-200PF

El Boeing 757-200 PF es una aeronave muy utilizada por las compañías de carga aérea exprés y de paquetería. La característica principal de éste carguero que lo diferencia de otros aviones de carga cómo el B747-8F, es que esta únicamente pensado para el transporte paletizado de la carga. Por lo que los procesos suelen ser más agiles y únicamente se trata un tipo de carga.

Esta aeronave tiene dos compartimentos de carga: La bodega principal y la bodega inferior. Con una capacidad de 15 pallets.

Los procesos o caminos que definen la escala en forma de red (Figura 17) del B757-200PF son los siguientes:

[1] Combustible.

[2] Mantenimiento.

[3] Descarga/Carga de bodega inferior.

[4] Descarga/Carga de bodega principal

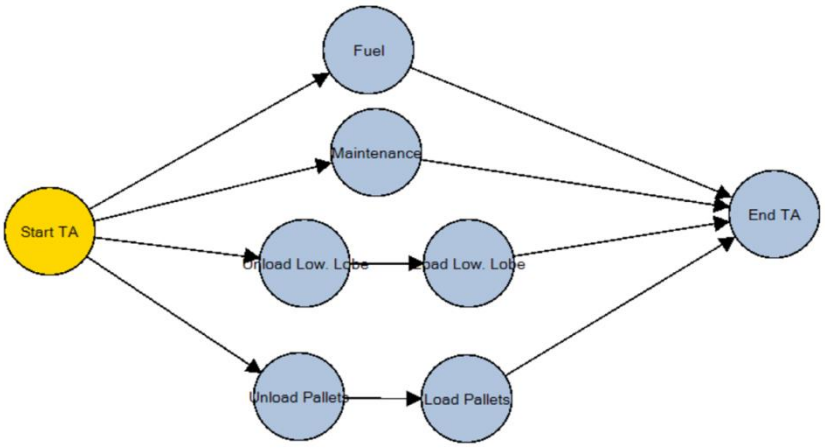


Figura 17 Representación de la escala del B757-200PF en forma de red

Los tiempos medios estimados de realización de las diferentes actividades (Tabla 13), se han basado en los tiempos descritos en el manual de la aeronave.

B757-200PF			
	Number	Activities	minutes
Cargo	1	Unload Pallets	25
	2	Load Pallets	25
	3	Unload Low Lobe	20
	4	Load Low Lobe	20
Aircraft Services	5	Fuel	15
	6	Manteinance	5

Tabla 13 Tiempos de las actividades de la escala del B757-200PF

Se puede observar en los outputs del algoritmo (Tabla 14) que sucede algo parecido al caso anteriormente estudiado, obtenemos que el camino más largo corresponde en prácticamente el 100% de los casos al camino de Descarga/Carga de la bodega principal (Camino 4). Por lo que igualmente que en el caso anterior, puede resultar más interesante conocer los caminos más cortos para que de esta forma se puedan optimizar los recursos y por tanto los costes.

B757-200PF		
Mean TA: 50 min.		
Paths	Largest	Shortest
path1	0,00%	0,02%
path2	0,00%	99,98%
path3	0,24%	0,00%
path4	99,76%	0,00%

Tabla 14 Outputs del T-CPA para el B757-200PF

9 PROPUESTA DE MEJORA

En este apartado se aporta una propuesta de mejora para una de las escalas definidas en el apartado anterior. El objetivo es el de demostrar las capacidades que ofrece el análisis que realiza el algoritmo y cómo pueden ser aplicadas en la escala de una aeronave concreta.

Se ha decidido analizar la escala del **Boeing 737-800** debido a que se trata de uno de los aviones más populares del mundo. Los outputs del T-CPA para esta aeronave mostraban que el camino 1 (Desembarque de pasajeros-Limpieza-Embarque de pasajeros) era el más largo prácticamente en todos los casos, aunque se disminuyera la ocupación.

De este análisis podemos extraer que todos aquellos esfuerzos que se centren en reducir el camino 1 provocarán directamente una reducción del tiempo total de la escala. Por ejemplo, las tareas de limpieza duran unos 11 minutos. Si se consiguen reducir esos minutos, por ejemplo a 6 minutos, el tiempo total de la escala se reducirá 5 minutos.

Aunque el análisis del camino más largo no es el único análisis que se puede extraer, si se observan los caminos más cortos también se puede optimizar la escala de una forma indirecta.

Los outputs del T-CPA mostraban que los caminos más cortos de la escala del B737-800 se los reparten los caminos 2 (Catering), 4 (Combustible) y 5 (Mantenimiento) con un 40%,40%,20% respectivamente. Esto indica que los equipos dedicados a estos caminos, terminan sus tareas antes de terminar la escala.

La utilidad que ofrece el T-CPA es que se pueden probar diferentes estructuras de la escala y observar los resultados que ofrecerían en la realidad. Estas pruebas serán útiles siempre y que no se rompa la secuencialidad requerida de las actividades.

Teniendo esto en cuenta se ha probado a realizar un cambio de estructura en la forma original de la escala del Boeing 737-800 (Figura 18). Esta nueva estructura tiene los siguientes caminos:

[1] Desembarque de pasajeros-Limpieza-Embarque de pasajeros.
[2] Catering.
[3] Descarga y Carga de equipajes y carga.
[4] Descarga de equipaje y carga-Combustible-Mantenimiento.

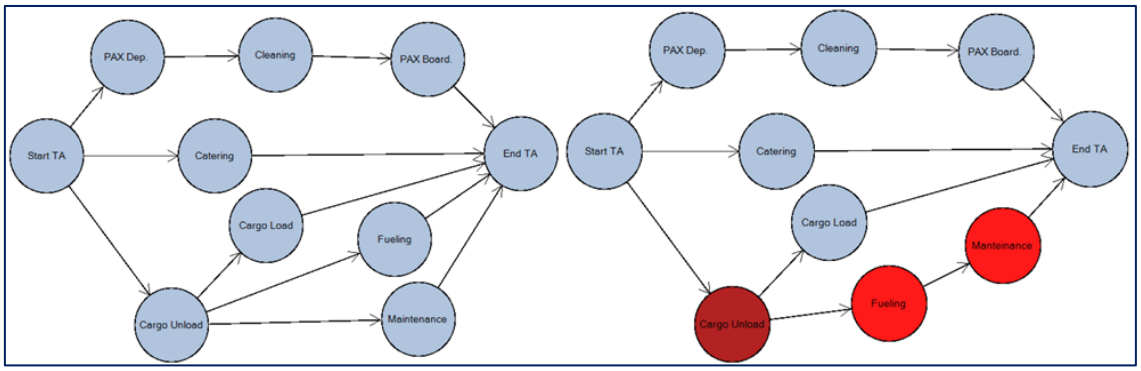


Figura 18 Original (Izquierda) y nueva (Derecha) estructuras de red de la escala del B737-800

Vemos que los caminos se han reducido en uno, por lo que pasa de tener 5 caminos a tener 4. Teniendo en cuenta los outputs del algoritmo, se ha probado a posicionar las tareas de mantenimiento para que se realicen una vez se ha terminado de cargar el combustible.

Este cambio se ha realizado porque ambas tareas se realizan en el exterior de la aeronave y por tanto no interfieren a otras tareas que se realizan en otras zonas.

Una vez definida la nueva estructura, se definen los inputs del algoritmo y se extraen los outputs de la nueva escala (Tabla 15). Para mejorar el análisis se han definido 3 escenarios diferentes, con un 100%, un 90% y un 75% de ocupación.

Original Structure								
Ocupación 100%			Ocupación 90%			Ocupación 75%		
Mean TA: 33,92 min.			Mean TA: 31 min			Mean TA: 28,1 min		
New Structure								
Ocupación 100%			Ocupación 90%			Ocupación 75%		
Mean TA: 33,96 min.			Mean TA: 31,11 min			Mean TA: 28,6 min		
Paths	Largest	Shortest	Paths	Largest	Shortest	Paths	Largest	Shortest
path1	98,50%	0,00%	path1	92,85%	0,00%	path1	76,79%	0,01%
path2	0,00%	99,00%	path2	0,00%	98,91%	path2	0,00%	98,98%
path3	0,02%	0,84%	path3	0,28%	0,93%	path3	1,35%	0,84%
path4	1,48%	0,16%	path4	6,87%	0,16%	path4	21,86%	0,17%

Tabla 15 Outputs T-CPA de la nueva escala del B737-800

Podemos observar que los tiempos medios de finalización de la escala son prácticamente los mismos, por tanto, a priori no supondría una desventaja el nuevo cambio de estructura.

Por otro lado se puede observar un cambio en la distribución de los caminos más cortos y más largos en función de la ocupación de la aeronave. Tal y cómo se puede observar en la Figura 19, la nueva estructura provoca que, aunque el camino 1 siga siendo el mayoritario en los tres escenarios, el camino número 4 (el nuevo camino) pasa a ser el camino más largo en un 20% de los casos aproximadamente.

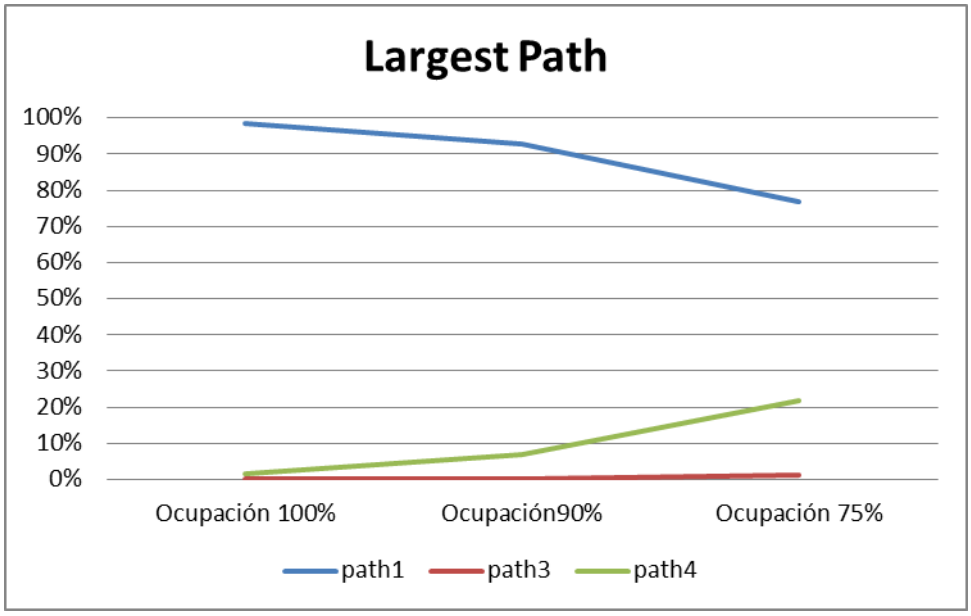


Figura 19 Distribución del camino más largo en la nueva estructura del B737-800

Podría parecer que aumenta el riesgo, ya que ahora son dos caminos los que pueden ser el más largo y por tanto el control puede ser más complicado. Pero hay que tener en cuenta que esta nueva estructura permite que se necesite un equipo humano menos (ya que el que se encarga del mantenimiento puede ocuparse también del combustible) y ese equipo puede destinarse a reforzar los caminos críticos disminuyendo la duración de sus actividades.

Sección 5: Conclusiones y trabajo futuro

10 CONCLUSIONES

La escala de las aeronaves es un proceso crítico para la actividad diaria de las aerolíneas, debido a que se trata de un período de tiempo en el que la aeronave esta parada y no produce beneficios a las aerolíneas. Por eso, un gran interés de las aerolíneas es el de reducir al máximo posible ese período de tiempo. Además, reducir el tiempo de escala permite dotar al aeropuerto con una mayor capacidad.

A partir de la revisión literaria de artículos de investigación sobre el tema, se ha podido detectar que existen dos formas de tratar el problema.

Por un lado se aborda el problema con la intención de conocer el comportamiento de cada una de las actividades que forman el proceso. Se busca conocer el comportamiento de las actividades para saber cómo han de ser tratadas y cuáles de ellas son las que forman cuellos de botella y por tanto, son críticas en el proceso. De esta forma saber sobre cuáles es más necesario centrar los esfuerzos. Por otro lado se entiende la escala como un bloque sólido al que se le añade un *buffer* de tiempo al final del proceso, el cuál absorbe todos aquellos posibles retrasos que se hayan podido generar en el desarrollo de la escala.

En este trabajo se ha optado por desarrollar una herramienta, centrada en el primer problema, que sea capaz de detectar aquellas actividades o secuencia de actividades que resultan críticas para la escala. La finalidad de este trabajo es ofrecer una herramienta de gestión y planificación de los recursos.

La herramienta desarrollada es un algoritmo en lenguaje de programación C++ que se denomina T-CPA (*Turnaround Critical Path Analysis*). El desarrollo se ha basado en un algoritmo de fiabilidad existente, el SREMS++, sobre el cual se han tenido que realizar múltiples cambios de código para adaptarlo a la finalidad deseada.

Se trata de un programa en el que se han de añadir las estructuras de la escala y unas distribuciones estadísticas que representen los tiempos de cada actividad para que no se trate de tiempos deterministas. En el caso de este proyecto se han tratado los tiempos con distribuciones estadísticas *Weibull*.

Para probar la funcionalidad del T-CPA, se han desarrollado diferentes pruebas. Estas pruebas han consistido en utilizar las escalas propuestas por los fabricantes de las aeronaves y analizar qué actividades de estos procesos son críticas. Para la realización de pruebas se han escogido, por un lado aeronaves comerciales de transporte de pasajeros (B737, A380 y A320) y por otro lado dos aeronaves típicas cargueras (B747-8F y B757-200PF).

En el caso de las aeronaves de transporte de pasajeros el análisis ha sido más exhaustivo, ya que la variabilidad que se puede dar en este tipo de escalas es mayor. Por un lado se han comparado dos aeronaves muy similares (B737 y A320), y por otro lado, para el B737 y el A380, se han definido diferentes escenarios de ocupación de pasajeros para conocer el comportamiento de los caminos críticos.

En el caso de la escala del B737 se ha planteado una mejora de la escala, que no reduce el tiempo de escala, pero sí que permite optimizar los recursos.

Los resultados obtenidos muestran que, en el caso de las aerolíneas de transporte de pasajeros, las actividades que están asociadas directamente con los pasajeros son críticas. Por tanto, es ahí donde tienen que centrar los esfuerzos las aerolíneas y compañías de *handling*.

Por otro lado, se ha ejemplificado la importancia de conocer aquellos caminos que son más cortos, ya que permiten al gestor reasignar aquellos recursos que quedan ociosos durante una parte de la escala. Por tanto, la herramienta dota de capacidad al gestor para reasignar esos recursos desaprovechados. Con el objetivo, por ejemplo, de mejorar las actividades que sí son críticas, tal y cómo se demuestra en la mejora propuesta de la escala del B737.

11 TRABAJO FUTURO

En este trabajo se ha presentado una primera versión del algoritmo T-CPA, capaz de detectar aquellas actividades y procesos que son críticos en la escala de las aeronaves, pudiendo utilizar cualquier tipo de aeronave siempre que se conozcan los tiempos de las actividades y su varianza.

Al tratarse de una primera versión, existen numerosas mejoras que se podrían realizar para crear un algoritmo con mejores resultados, más intuitivos y más flexibles a las diferentes situaciones que pueden darse en el aeropuerto. A continuación se definen posibles mejoras que se podrían añadir al modelo aquí presentado:

- Una mejora que resultaría muy interesante sería añadir al algoritmo cierta flexibilidad para adaptarlo a las situaciones reales que se dan en los aeropuertos. Es decir, tener en cuenta factores que puedan hacer que los tiempos medios de las actividades varíen de otra forma a la estándar. Por ejemplo, tener en cuenta el tipo de aeropuerto en el que estamos, la congestión que hay normalmente en ese aeropuerto o añadir las posibles variaciones que puede tener la escala en situaciones de meteorología extrema.
- Una buena forma de validar este algoritmo seria implementarlo en otros programas de simulación. Además, implementar los experimentos realizados en otros programas, sería una buena forma de comparar resultados, como por ejemplo con los programas de simulación SIMIO o CAST.
- Este algoritmo ofrece resultados basados únicamente en términos de tiempo. Una mejora del T-CPA sería añadir el factor de coste al análisis, ofreciendo así una mejor herramienta para el gestor. Al fin y al cabo lo que buscan las aerolíneas es reducir el coste de la escala por lo que podría ser muy interesante dicha mejora.
- Los experimentos y pruebas realizados en este trabajo, se han basado en los tiempos que ofrecen las constructoras de aeronaves. Estos tiempos son puramente orientativos, ya que cada aerolínea tiene diferentes distribuciones de asientos y los diferentes procesos pueden no ser idénticos a los que propone la constructora. Sería muy interesante implementar el algoritmo con los datos de diferentes tipos de aerolíneas.
- Mejorar el algoritmo en su aspecto más visual e intuitivo. Hacer el programa más atractivo para el usuario creando, por ejemplo, una interfaz más visual para el usuario y unos outputs más intuitivos para el análisis.

12 REFERENCIAS BIBLIOGRÁFICAS

Airbus S.A.S., (2005) *A320/A320NEO, Aircraft Characteristics Airport and Maintenance Planning*, Technical Data Support and Services.

Airbus S.A.S., (2005) *A380, Aircraft Characteristics Airport and Maintenance Planning*, Technical Data Support and Services.

Boeing, (2005), *737 Airplane Characteristics for Airport Planning*, Boeing Commercial Airplanes.

Boeing, (2012), *747 Airplane Characteristics for Airport Planning*, Boeing Commercial Airplanes.

Boeing, (1999), *757-200/300 Airplane Characteristics for Airport Planning*, Boeing Commercial Airplanes.

B. Oreschko, T.Kunze, M. Schultz, H. Fricke, V.Kumar and L. Sherry (2012). *Turnaround Prediction with Stochastic Process Times and Airport Specific Delay Pattern*, Best Paper in Track - International Conference on Research in Air Transportation, Berkeley (USA).

CODA Digest, (2013), *Delays to Air Transport in Europe-Annual 2013*, CODA, Eurocontrol, Brussels.

Cook A., Tanner G., Anderson S., (2004) *Evaluating the true cost to airlines of one minute of airborne or ground delay*, Performance Review Commission, University of Westminster.

Fricke H., Schultz M., (2009) *Delay Impacts onto Turnaround Performance*, USA -Europe Air Traffic Management Research and Development Seminar.

Kuster J., Jannach D., Friedrich G., (2008) *Extending the RCPSP for modeling and solving disruption management problems,* Springer Science + Business Media.

Makhloof M. A. A., Waheed M. E. (2014). *Optimization for Aircraft Turnaround Operations using Project Evaluation and Review Technique*. International Journal of Information Technology & Decision Making.

Wu C.L., (2010) *Airline Operations and Delay Management: Insights from Airline Economics, Networks and Strategic Schedule Planning*, Ashgate.

Wu, C., & Caves, R. E. (2004). *Modelling and simulation of aircraft turnaround operations at airports.* Transportation Planning and Technology, 27(1), 25–46.

Wu C.L., Caves R.E., (2010) *Flight schedule punctuality control and management: a stochastic approach*, Routledge.

Anexos

A continuación se muestra el código del T-CPA, desarrollado en C++ en el Visual Studio 2010.

- **T-CPA.h**

En el siguiente código, se definen las estructuras que se van a utilizar para la simulación.

```
const char firstFile[] = "T-CPA_inputs_first.txt";
const char secondFile[] = "T-CPA_inputs_second.txt";
const char thirdFile[] = "T-CPA_inputs_third.txt";
const char outputsFile[] = "T-CPA_outputs.txt";

struct simInputs
{
	// First-level inputs (get from file "FIRST.TXT")
	unsigned long int nIterations;
	unsigned short int nComponents;
	unsigned short int nTargetTimes;
	float timeBetweenTargets;
	unsigned short int nPaths;
	int seed;

	// Second-level inputs (get from file "SECOND.TXT")
	// Failure-time distribution for each component and length of each path
	unsigned char *ftDistribution;
	float *ftdParameter1;
	float *ftdParameter2;
	unsigned short int *pathLength;

	// Third-level inputs (get from file "THIRD.TXT")
	// Composition of each path (system structure defined from minimal paths)
	unsigned short int **pathComposition;
};

struct iterationOutputs
{
	double systemFailureTime;
	unsigned short int LargestPath;
	unsigned short int ShortestPath;
	unsigned short int keyCompInSP;
	unsigned short int keyCompInWP;
};

struct simOutputs
{
	// simulation length
	time_t simLength;

	// systemFailureTime statistics
	double mean_SFT;
	double stDev_SFT;
	double left_99CI_mean_SFT;
	double right_99CI_mean_SFT;

	// system reliability for each target time
	double *systemReliability;
	double *left_99CI_SR;
	double *right_99CI_SR;

	// strongest and weakest paths frequencies
	unsigned long *LPfrequency;
	unsigned long *SPfrequency;
```

```
	// key components in SP and WP frequencies
	unsigned long *KCinSPfrequency;
	unsigned long *KCinWPfrequency;

	double *MinDuration;
	double *MaxDuration;
	double *MeanDuration;

	double *MinDurationComp;
	double *MaxDurationComp;
	double *MeanDurationComp;

	//
	double **PathsDuration;
	double **CompDuration;

};


/*************************************************************************
		functions declaration (prototypes)
*************************************************************************/

void getFirstLevelData(struct simInputs *p_si);
void getSecondLevelData(struct simInputs *p_si);
void getThirdLevelData(struct simInputs *p_si);
void performSimulation(struct simInputs *p_si, struct iterationOutputs *p_io,
struct simOutputs *p_so);
void calcSimOutputs(struct simInputs *p_si, struct iterationOutputs *p_io,
                                struct simOutputs *p_so);
void reportSummary(struct simInputs *p_si, struct simOutputs *p_so);
```

- **T-CPA.cpp**

A continuación se muestra el código principal del programa en el que se realizan los pasos principales. También se muestran las funciones que cogen los datos de los ficheros de texto (getFirstLevelData, getSecondLevelData y getThirdLevelData).

```
void main()
{
	time_t startTime, endTime;
	unsigned short path;
	struct simInputs si, *p_si = &si;
	struct simOutputs so, *p_so = &so;
	struct iterationOutputs *p_io = NULL;

	system("cls");
	printf("\nINITIALIZING VARIABLES, PLEASE WAIT...");


// 1. GET SIMULATION INPUTS


	// Get first-level inputs
	getFirstLevelData(p_si);

	// Using first-level inputs, redim second-level inputs
```

```
	p_si->ftDistribution = new unsigned char[p_si->nComponents];
	p_si->ftdParameter1 = new float[p_si->nComponents];
	p_si->ftdParameter2 = new float[p_si->nComponents];
	p_si->pathLength = new unsigned short int[p_si->nPaths];

	// Get second-level inputs
	getSecondLevelData(p_si);

	// Using second-level inputs, redim third-level inputs
	p_si->pathComposition = new unsigned short int*[p_si->nPaths];
	for(path = 0; path < p_si->nPaths; path++)
		p_si->pathComposition[path] = new unsigned short int[p_si-
>pathLength[path]];


	// Get third-level inputs
	getThirdLevelData(p_si);


// 2. REDIM ITERATION OUTPUTS AND SIMULATION OUTPUTS


	// Redim iterationOutputs once the nIterations is known

	p_io = new struct iterationOutputs[p_si->nIterations];



	// Redim simulationOutputs once the nTargetTimes is known
	p_so->systemReliability = new double[p_si->nTargetTimes];
	p_so->left_99CI_SR = new double[p_si->nTargetTimes];
	p_so->right_99CI_SR = new double[p_si->nTargetTimes];
	p_so->LPfrequency = new unsigned long[p_si->nPaths];
	p_so->SPfrequency = new unsigned long[p_si->nPaths];
	p_so->KCinSPfrequency = new unsigned long[p_si->nComponents];
	p_so->KCinWPfrequency = new unsigned long[p_si->nComponents];
	p_so->MaxDuration = new double[p_si->nPaths];
	p_so->MinDuration = new double[p_si->nPaths];
	p_so->MeanDuration = new double [p_si->nPaths];
	p_so->MaxDurationComp = new double[p_si->nComponents];
	p_so->MinDurationComp = new double[p_si->nComponents];
	p_so->MeanDurationComp = new double [p_si->nComponents];


	p_so->PathsDuration = new double*[p_si->nIterations];
	for(int iteration=0;iteration<p_si->nIterations;iteration++)
		p_so->PathsDuration[iteration]= new double [p_si->nPaths];
	p_so->CompDuration = new double*[p_si->nIterations];
	for(int iteration=0;iteration<p_si->nIterations;iteration++)
		p_so->CompDuration[iteration]= new double [p_si->nComponents];


// 3. PERFORM SIMULATION


	startTime = time(0);
	performSimulation(p_si, p_io, p_so);
	endTime = time(0);
	p_so->simLength = endTime - startTime;
```

```
// 4. CALCULATE SIMULATION OUTPUTS

	calcSimOutputs(p_si, p_io, p_so);

// 5. FREE MEMORY
	delete[] p_io;
	p_io = NULL;

// 6. REPORT A SUMMARY

	reportSummary(p_si, p_so);
}


void getFirstLevelData(struct simInputs *p_si)
{
	/* First-level inputs:
		nIterations, nComponents, nTargetTimes, timeBetweenTargets,
		nPaths, seed */

	FILE *fp;


	if ( (fp = fopen(firstFile, "r")) == NULL)
	{
		printf("Error opening file: %s", firstFile);
		exit(1);
	}

	fscanf(fp, "%d %ld %ld %f %ld %d", &p_si->nIterations, &p_si->nComponents,
		&p_si->nTargetTimes, &p_si->timeBetweenTargets, &p_si->nPaths,
&p_si->seed);

	fclose(fp);

	return;
}



void getSecondLevelData(struct simInputs *p_si)
{
	/* Second-level inputs:
		ftDistribution[], ftdParameter1[], ftdParameter2[], pathLength[] */

	FILE *fp;
	unsigned short int comp, path;

	if ( (fp = fopen(secondFile, "r")) == NULL)
	{
		printf("Error opening file: %s", secondFile);
		exit(1);
	}

	for (comp = 0; comp < p_si->nComponents; comp++)
	{
		fscanf(fp,"%s", &p_si->ftDistribution[comp]);
		fflush(stdin);
		fscanf(fp,"%f %f", &p_si->ftdParameter1[comp], &p_si-
>ftdParameter2[comp]);
```

```
	}

	for (path = 0; path < p_si->nPaths; path++)
	{
		fscanf(fp,"%d", &p_si->pathLength[path]);
	}

	fclose(fp);

	return;
}


void getThirdLevelData(struct simInputs *p_si)
{

	/* Third-level inputs:
		pathComposition[][] */

	FILE *fp;
	unsigned short path, pos;

	if ( ( fp = fopen( thirdFile, "r" ) ) == NULL )
	{
		printf("Error opening file: %s", thirdFile);
		exit(1);
	}

	for (path = 0; path < p_si->nPaths; path++)
	{
		for (pos = 0; pos < p_si->pathLength[path]; pos++)
		{
			fscanf(fp,"%d", &p_si->pathComposition[path][pos]);

			// Caution!: components from file will go from 1 to
nComponents, while
			//     components inside the program will go from 0 to
nComponents - 1
			p_si->pathComposition[path][pos]--;
		}
	}

	fclose(fp);

	return;
}
```

- **performSimulation.cpp**

El siguiente código realiza la simulación de las diferentes iteraciones y almacena los datos obtenidos.

```
void performSimulation(struct simInputs *p_si, struct iterationOutputs *p_io,
struct simOutputs *p_so)
{
unsigned long int iteration;
unsigned short int comp, path, position;
```

```
unsigned short int ShortestPath, LargestPath;
double percentDone, progressBar, step = 0.10;
unsigned short *keyComp = NULL; // key component in each path
double *compFT = NULL; //time for each component
double *pathFT = NULL; //time for each path
double *PathDuration = NULL;


PathDuration= new double[p_si->nPaths];



FOR EACH ITERATION, GENERATE COMPONENTS FT AND CALCULATE ITER OUTPUTS


progressBar = 0.0;

for (iteration = 0; iteration < p_si->nIterations; iteration++)
{
       compFT = new double[p_si->nComponents];
       // Check if user information has to be updated
       percentDone = (double) iteration / p_si->nIterations;
       if(percentDone > progressBar + step)
              {
              progressBar += step;
              informUserOnSimProgress(progressBar);
              }

// Generate a random time for each component
for (comp = 0; comp < p_si->nComponents; comp++)
{

compFT[comp] = randomVariates( p_si->ftDistribution[comp], p_si-
>ftdParameter1[comp], p_si->ftdParameter2[comp], p_si->seed );

p_so->CompDuration[iteration][comp] = compFT[comp];}
// Calculate iteration outputs
LargestPath = 0;
ShortestPath = 0;
pathFT = new double[p_si->nPaths];
PathDuration = new double [p_si->nPaths];

for (path = 0; path < p_si->nPaths; path++)
{
for (position = 0; position < p_si->pathLength[path]; position++)
       {

if(pathFT[path]<=0)
       pathFT[path] = compFT[p_si->pathComposition[path][position]];

else
       pathFT[path] = pathFT[path] + compFT[p_si-
>pathComposition[path][position]];
       }

if(pathFT[path] < pathFT[ShortestPath] ) ShortestPath = path;

if(pathFT[path] > pathFT[LargestPath] ) LargestPath = path;

// Save time for each iterarion & path
```

```
p_so->PathsDuration[iteration][path]=pathFT[path];          }

p_io[iteration].LargestPath = LargestPath;
p_io[iteration].ShortestPath = ShortestPath;
p_io[iteration].systemFailureTime = pathFT[LargestPath];


delete[] pathFT;
pathFT = NULL;

delete[] PathDuration;
PathDuration = NULL;

delete[] compFT;
compFT = NULL;

       }
       return;
}
```

- **calcSimOutputs.cpp**

El siguiente código calcula los outputs del programa.

```
void calcSystemFTStats(struct simInputs *p_si, struct iterationOutputs *p_io,
struct simOutputs *p_so)
{

unsigned long int iteration;
double error99CI;

double *SFT = NULL;
SFT = new double[p_si->nIterations];

for (iteration = 0; iteration < p_si->nIterations; iteration++)
{
SFT[iteration] = p_io[iteration].systemFailureTime;
}

/* Use stats functions to obtain statistics. */

p_so->mean_SFT = mean(SFT, p_si->nIterations);
p_so->stDev_SFT = adjStDev(SFT, p_si->nIterations);
error99CI = tCIerror(0.99, p_so->stDev_SFT, p_si->nIterations);
p_so->left_99CI_mean_SFT = p_so->mean_SFT - error99CI;
p_so->right_99CI_mean_SFT = p_so->mean_SFT + error99CI;




//////////////////////
//STATISTICS OF PATHS//
//////////////////////

for (int path=0;path<p_si->nPaths;path++)
{
double *DT = NULL;
       DT = new double[p_si->nIterations];
       for(int iteration = 0;iteration<p_si->nIterations;iteration++)
```

```
{
		DT[iteration]=p_so->PathsDuration[iteration][path];
	}
	p_so->MaxDuration[path]= max(DT,p_si->nIterations);

delete[] DT;
	DT = NULL;
}

for (int path=0;path<p_si->nPaths;path++)
{
	double *DT = NULL;
	DT = new double[p_si->nIterations];
	for(int iteration = 0;iteration<p_si->nIterations;iteration++)
{
		DT[iteration]=p_so->PathsDuration[iteration][path];
	}
	p_so->MinDuration[path]= min(DT,p_si->nIterations);

delete[] DT;
	DT = NULL;
}

for (int path=0;path<p_si->nPaths;path++)
{
	double *DT = NULL;
	DT = new double[p_si->nIterations];
	for(int iteration = 0;iteration<p_si->nIterations;iteration++)
{
		DT[iteration]=p_so->PathsDuration[iteration][path];
	}
	p_so->MeanDuration[path]= mean(DT,p_si->nIterations);

delete[] DT;
	DT = NULL;
}
////////////////////////////
//STATISTICS OF ACTIVITIES//
////////////////////////////

for (int comp=0;comp<p_si->nComponents;comp++)
{
	double *DT = NULL;
	DT = new double[p_si->nIterations];
	for(int iteration = 0;iteration<p_si->nIterations;iteration++)
{
		DT[iteration]=p_so->CompDuration[iteration][comp];
	}
	p_so->MaxDurationComp[comp]= max(DT,p_si->nIterations);

delete[] DT;
	DT = NULL;
}

for (int comp=0;comp<p_si->nComponents;comp++)
{
	double *DT = NULL;
	DT = new double[p_si->nIterations];
	for(int iteration = 0;iteration<p_si->nIterations;iteration++)
{
		DT[iteration]=p_so->CompDuration[iteration][comp];
```

```
	}
	p_so->MinDurationComp[comp]= min(DT,p_si->nIterations);

delete[] DT;
	DT = NULL;
}

for (int comp=0;comp<p_si->nComponents;comp++)
{
	double *DT = NULL;
	DT = new double[p_si->nIterations];
	for(int iteration = 0;iteration<p_si->nIterations;iteration++)
{
		DT[iteration]=p_so->CompDuration[iteration][comp];
	}
	p_so->MeanDurationComp[comp]= mean(DT,p_si->nIterations);

delete[] DT;
	DT = NULL;
}


	/* Free memory. */
	delete[] SFT;
	SFT = NULL;

	return;
}


/*****************************************************************************
			calcSystemReliability() function
*****************************************************************************/

void calcSystemReliability(struct simInputs *p_si, struct iterationOutputs *p_io,
					struct simOutputs *p_so)

{
	unsigned short int target;
	unsigned long int iteration;
	float targetTime;
	double error99CI;
	double p; // estimated probability of success
	unsigned long int *success;

	/* Redim success array. */
	success = new unsigned long int[p_si->nTargetTimes];

	/* Initialize success array. */
	for (target = 0; target < p_si->nTargetTimes; target++)
	{
		success[target] = 0;
	}

	/* For each iteration, assign a "success" to those target times that are
		smaller than the obtained systemFailureTime. */
	for (iteration = 0; iteration < p_si->nIterations; iteration++)
	{
		targetTime = 0;
		for (target = 0; target < p_si->nTargetTimes; target++)
		{
```

```
                        targetTime = targetTime + p_si->timeBetweenTargets;
                        if (p_io[iteration].systemFailureTime > targetTime)
                                success[target]++;
                }
        }

        /* For each target time, calculate its associated reliability. */
        for (target = 0; target < p_si->nTargetTimes; target++)
        {
                p = (double) success[target] / p_si->nIterations;
                p_so->systemReliability[target] = p;

                error99CI = zCIerror(0.99, sqrt(p*(1-p)), p_si->nIterations);
                p_so->left_99CI_SR[target] = p - error99CI;
                p_so->right_99CI_SR[target] = p + error99CI;
        }

        return;
}


/*****************************************************************************
                            calcKeyCompStats() function
*****************************************************************************/

void calcKeyCompStats(struct simInputs *p_si, struct iterationOutputs *p_io,
                                        struct simOutputs *p_so)
{

        unsigned long iteration;
        unsigned short path;
        unsigned short comp;

        /* Initialize LP and SP frequencies. */
        for(path = 0; path < p_si->nPaths; path++)
        {
                p_so->LPfrequency[path] = 0;
                p_so->SPfrequency[path] = 0;
        }

        /* Initialize KC in LP and KC in SP frequencies. */
        for(comp = 0; comp < p_si->nComponents; comp++)
        {
                p_so->KCinSPfrequency[comp] = 0;
                p_so->KCinWPfrequency[comp] = 0;
        }

        /* Calculate frequencies. */
        for(iteration = 0; iteration < p_si->nIterations; iteration++)
        {
                /* Frequencies for each LP and SP path. */
                for(path = 0; path < p_si->nPaths; path++)
                {
                        if(path == p_io[iteration].LargestPath)
                                p_so->LPfrequency[path]++;
                        if(path == p_io[iteration].ShortestPath)
                                p_so->SPfrequency[path]++;
                }

                /* Frequencies for each KC in SP and KC in WP. */
                for(comp = 0; comp < p_si->nComponents; comp++)
```

```
		{
			if(comp == p_io[iteration].keyCompInSP)
				p_so->KCinSPfrequency[comp]++;
			if(comp == p_io[iteration].keyCompInWP)
				p_so->KCinWPfrequency[comp]++;
		}
	}

	return;
}
```

- **reportSummary.cpp**

```
void reportSummary(struct simInputs *p_si, struct simOutputs *p_so)
{
	FILE *fp;

	unsigned short int comp, path, target, pos;

	if ( ( fp = fopen( outputsFile, "w" ) ) == NULL )
	{
		printf("Error opening file: %s", outputsFile);
		exit(1);
	}


	// Inputs from firstFile
fprintf(fp, "INPUTS FROM: %s", firstFile);
fprintf(fp, "\n%d %ld %ld %f %ld %d", p_si->nIterations, p_si->nComponents,
	p_si->nTargetTimes, p_si->timeBetweenTargets, p_si->nPaths, p_si->seed);

fprintf(fp, "\n\n");


	// Inputs from secondFile
fprintf(fp, "INPUTS FROM: %s\n", secondFile);

for (comp = 0; comp < p_si->nComponents; comp++)
{
	fprintf(fp, "%c %f %f", p_si->ftDistribution[comp],
		p_si->ftdParameter1[comp], p_si->ftdParameter2[comp]);

fprintf(fp,"\n");
}

fprintf(fp,"\n");

for (path = 0; path < p_si->nPaths; path++)
{
	fprintf(fp,"\n%d", p_si->pathLength[path]);
}

fprintf(fp, "\n\n");



	// Inputs from thirdFile
fprintf(fp, "INPUTS FROM: %s\n", thirdFile);

for (path = 0; path < p_si->nPaths; path++)
```

```
{
	fprintf(fp, "\n%d:  ", path + 1);

	for (pos = 0; pos < p_si->pathLength[path]; pos++)
	{
		fprintf(fp, " %d", p_si->pathComposition[path][pos] + 1);
	}
}

fprintf(fp, "\n\n");


	// Simulation outputs
fprintf(fp, "SIMULATION OUTPUTS:\n");


fprintf(fp, "%ld ",p_so->simLength);
fprintf(fp, "%lf %lf %lf %lf", p_so->mean_SFT, p_so->stDev_SFT,
		p_so->left_99CI_mean_SFT, p_so->right_99CI_mean_SFT);
fprintf(fp,"\n");


for (target = 0; target < p_si->nTargetTimes; target++)
{
fprintf(fp, "\n%d:   %lf %lf %lf", target + 1, (p_so-
>systemReliability[target])*100,(p_so->left_99CI_SR[target])*100,(p_so-
>right_99CI_SR[target])*100);
}

fprintf(fp, "\n");
fprintf(fp, "\nPath  Largest      Shortest   MIN        MAX   MEAN");

for (path = 0; path < p_si->nPaths; path++)
{
fprintf(fp,"\n%d:   %.2f   %.2f    %.3f  %.3f   %.3f", path +1,(p_so-
>LPfrequency[path]/(double) p_si->nIterations)*100,(p_so-
>SPfrequency[path]/(double) p_si->nIterations)*100,p_so->MinDuration[path],p_so-
>MaxDuration[path],p_so->MeanDuration[path]);
}

fprintf(fp, "\n");
fprintf(fp, "\nActivity  MIN          MAX           MEAN");

for(int comp=0; comp<p_si->nComponents; comp++)
{
fprintf(fp,"\n%d:    | %.3f | %.3f | %.3f",comp+1,p_so-
>MinDurationComp[comp],p_so->MaxDurationComp[comp], p_so-
>MeanDurationComp[comp]);
}

fclose(fp);

informUserOnEndOfSim(p_so);

return;
}
```

- **Resultados de las simulaciones**

A continuación se muestran todos los outputs obtenidos en las simulaciones realizadas para probar la herramienta.

- ***A320-200***

INPUTS FROM: T-CPA_inputs_first.txt
50000 8 10 5.000000 5 30384

INPUTS FROM: T-CPA_inputs_second.txt
w 5.543100 8.661600
w 9.344100 13.705000
w 10.108000 14.709200
w 7.058300 10.685600
w 14.702200 20.724300
w 8.572800 12.689200
w 4.042000 6.615600
w 5.534500 8.650100

3
3
2
1
2

INPUTS FROM: T-CPA_inputs_third.txt

1: 1 3 2
2: 1 4 2
3: 1 5
4: 6
5: 7 8

SIMULATION OUTPUTS:
1 35.039250 2.815143 35.006820 35.071680

1: 100.000000 100.000000 100.000000
2: 100.000000 100.000000 100.000000
3: 100.000000 100.000000 100.000000
4: 100.000000 100.000000 100.000000
5: 99.970000 99.950051 99.989949
6: 95.724000 95.490943 95.957057
7: 52.246000 51.670608 52.821392
8: 3.160000 2.958487 3.361513
9: 0.000000 0.000000 0.000000
10: 0.000000 0.000000 0.000000

Path	Largest	Shortest	MIN	MAX	MEAN
1:	94.57	0.00	21.114	44.503	34.978
2:	4.59	0.00	16.328	41.180	30.974
3:	0.84	0.00	16.484	35.971	28.003
4:	0.00	75.61	3.340	16.910	11.989
5:	0.00	24.39	4.471	22.876	13.998

Activity	MIN	MAX	MEAN
1:	1.090	13.785	7.988
2:	3.813	17.759	12.997
3:	3.320	18.716	13.993
4:	2.418	14.962	9.989
5:	9.677	24.172	20.015
6:	3.340	16.910	11.989
7:	0.515	11.947	6.003
8:	1.501	13.162	7.996

- ***B737-800[100% de Ocupación]***

INPUTS FROM: T-CPA_inputs_first.txt
50000 8 10 5.000000 5 28650

INPUTS FROM: T-CPA_inputs_second.txt
w 6.249800 9.609000
w 10.103300 14.703100
w 10.869300 15.713000
w 7.818900 11.693500
w 4.045100 6.615400
w 11.634300 16.716000
w 6.249800 9.609000
w 7.058300 10.685600

3
1
2
2
2

INPUTS FROM: T-CPA_inputs_third.txt

1: 1 4 2
2: 3
3: 5 6
4: 5 7
5: 5 8

SIMULATION OUTPUTS:
1 33.924242 2.887964 33.890973 33.957512

1: 100.000000 100.000000 100.000000
2: 100.000000 100.000000 100.000000
3: 100.000000 100.000000 100.000000
4: 100.000000 100.000000 100.000000
5: 99.746000 99.688018 99.803982
6: 90.658000 90.322761 90.993239
7: 37.068000 36.511625 37.624375
8: 1.010000 0.894817 1.125183
9: 0.000000 0.000000 0.000000
10: 0.000000 0.000000 0.000000

Path	Largest	Shortest	MIN	MAX	MEAN
1:	99.91	0.00	20.236	44.146	33.923
2:	0.00	41.70	6.446	19.715	14.996
3:	0.09	0.06	9.993	29.826	22.006
4:	0.00	40.12	5.090	23.135	14.929
5:	0.00	18.12	5.366	24.085	15.992

Activity	MIN	MAX	MEAN
1:	1.568	13.984	8.940
2:	4.522	18.596	13.986
3:	6.446	19.715	14.996
4:	3.200	16.144	10.998
5:	0.427	12.588	6.001
6:	6.192	20.572	16.004
7:	1.642	14.414	8.928
8:	2.228	14.820	9.991

- ***B737-800[90% de Ocupación]***

INPUTS FROM: T-CPA_inputs_first.txt
50000 8 10 5.000000 5 11617

INPUTS FROM: T-CPA_inputs_second.txt
w 5.543100 8.661700
w 8.581000 12.699800
w 10.869300 15.713000
w 7.818900 11.693500
w 4.045100 6.615400
w 11.634300 16.716000
w 6.249800 9.609000
w 7.058300 10.685600

3
1
2
2
2

INPUTS FROM: T-CPA_inputs_third.txt

1: 1 4 2
2: 3
3: 5 6
4: 5 7
5: 5 8

SIMULATION OUTPUTS:
1 31.010746 2.844310 30.977980 31.043512

1: 100.000000 100.000000 100.000000
2: 100.000000 100.000000 100.000000
3: 100.000000 100.000000 100.000000
4: 99.994000 99.985077 100.002923
5: 97.802000 97.633104 97.970896
6: 65.038000 64.488695 65.587305
7: 7.480000 7.176959 7.783041
8: 0.010000 -0.001519 0.021519
9: 0.000000 0.000000 0.000000
10: 0.000000 0.000000 0.000000

Path	Largest	Shortest	MIN	MAX	MEAN
1:	99.08	0.00	18.661	41.138	30.999
2:	0.00	41.82	5.496	19.613	14.994
3:	0.92	0.08	10.750	29.908	22.015
4:	0.00	40.03	5.167	23.174	14.929
5:	0.00	18.07	5.879	26.073	16.010

Activity	MIN	MAX	MEAN
1:	1.167	13.408	8.011
2:	3.953	16.963	11.996
3:	5.496	19.613	14.994
4:	2.873	15.724	10.992
5:	0.394	11.638	6.009
6:	6.993	20.766	16.006
7:	1.111	14.393	8.919
8:	1.954	14.857	10.001

- ***B737-800[75% de Ocupación]***

INPUTS FROM: T-CPA_inputs_first.txt
50000 8 10 5.000000 5 31319

INPUTS FROM: T-CPA_inputs_second.txt
w 4.790200 7.642800
w 7.058300 10.685600
w 10.869300 15.713000
w 7.818900 11.693500
w 4.045100 6.615400
w 11.634300 16.716000
w 6.249800 9.609000
w 7.058300 10.685600

3
1
2
2
2

INPUTS FROM: T-CPA_inputs_third.txt

1: 1 4 2
2: 3
3: 5 6
4: 5 7
5: 5 8

SIMULATION OUTPUTS:
1 28.067716 2.725817 28.036315 28.099117

1: 100.000000 100.000000 100.000000
2: 100.000000 100.000000 100.000000
3: 100.000000 100.000000 100.000000
4: 99.902000 99.865956 99.938044
5: 86.520000 86.126599 86.913401
6: 24.848000 24.350208 25.345792
7: 0.346000 0.278358 0.413642
8: 0.000000 0.000000 0.000000
9: 0.000000 0.000000 0.000000
10: 0.000000 0.000000 0.000000

Path	Largest	Shortest	MIN	MAX	MEAN
1:	94.66	0.00	15.418	39.545	27.982
2:	0.00	41.65	6.312	19.610	15.002
3:	5.34	0.05	11.303	30.864	22.003
4:	0.00	40.38	5.822	23.308	14.936
5:	0.00	17.92	5.304	23.858	16.000

Activity	MIN	MAX	MEAN
1:	0.408	12.875	6.993
2:	1.376	15.262	9.992
3:	6.312	19.610	15.002
4:	3.377	15.861	10.997
5:	0.267	11.716	6.005
6:	6.668	20.434	15.998
7:	1.868	14.048	8.931
8:	2.000	14.867	9.996

- ***A380-800[100% de Ocupación]***

INPUTS FROM: T-CPA_inputs_first.txt
50000 8 20 5.000000 5 8376

INPUTS FROM: T-CPA_inputs_second.txt
w 11.637900 16.715799
w 18.539101 25.730499
w 33.905899 45.740398
w 22.380800 30.734301
w 18.539101 25.730499
w 22.378599 30.734400
w 33.905899 45.740398
w 45.438900 60.743099

3
3
2
1
2

INPUTS FROM: T-CPA_inputs_third.txt

1: 1 3 2
2: 1 4 2
3: 1 7
4: 8
5: 5 6

SIMULATION OUTPUTS:
1 85.999084 2.887645 85.965819 86.032350

1: 100.000000 100.000000 100.000000
2: 100.000000 100.000000 100.000000
3: 100.000000 100.000000 100.000000
4: 100.000000 100.000000 100.000000
5: 100.000000 100.000000 100.000000
6: 100.000000 100.000000 100.000000
7: 100.000000 100.000000 100.000000
8: 100.000000 100.000000 100.000000
9: 100.000000 100.000000 100.000000
10: 100.000000 100.000000 100.000000
11: 100.000000 100.000000 100.000000
12: 100.000000 100.000000 100.000000
13: 100.000000 100.000000 100.000000
14: 99.998000 99.992848 100.003152
15: 99.908000 99.873076 99.942924
16: 97.014000 96.817938 97.210062
17: 66.100000 65.554704 66.645296
18: 6.714000 6.425709 7.002291
19: 0.000000 0.000000 0.000000
20: 0.000000 0.000000 0.000000

Path	Largest	Shortest	MIN	MAX	MEAN
1:	100.00	0.00	69.695	94.706	85.999
2:	0.00	0.00	56.046	80.111	71.011
3:	0.00	3.48	46.085	68.082	61.007
4:	0.00	3.47	47.613	63.852	60.003
5:	0.00	93.04	39.807	62.185	54.989

Activity	MIN	MAX	MEAN
1:	6.679	20.582	16.004
2:	13.343	29.469	25.002
3:	32.866	48.924	44.994
4:	19.302	34.239	30.006
5:	14.329	29.246	24.997
6:	18.549	34.237	29.992
7:	34.107	49.156	45.003
8:	47.613	63.852	60.003

- ***A380-800[90% de Ocupación]***

INPUTS FROM: T-CPA_inputs_first.txt
50000 8 20 5.000000 5 19135

INPUTS FROM: T-CPA_inputs_second.txt
w 9.344000 13.704900
w 16.236799 22.727200
w 33.905899 45.740398
w 22.380800 30.734301
w 18.539101 25.730499
w 22.378599 30.734400
w 33.905899 45.740398
w 45.438900 60.743099

3
3
2
1
2

INPUTS FROM: T-CPA_inputs_third.txt

1: 1 3 2
2: 1 4 2
3: 1 7
4: 8
5: 5 6

SIMULATION OUTPUTS:
0 80.001867 2.884025 79.968644 80.035091

1: 100.000000 100.000000 100.000000
2: 100.000000 100.000000 100.000000
3: 100.000000 100.000000 100.000000
4: 100.000000 100.000000 100.000000
5: 100.000000 100.000000 100.000000
6: 100.000000 100.000000 100.000000
7: 100.000000 100.000000 100.000000
8: 100.000000 100.000000 100.000000
9: 100.000000 100.000000 100.000000
10: 100.000000 100.000000 100.000000
11: 100.000000 100.000000 100.000000
12: 100.000000 100.000000 100.000000
13: 99.998000 99.992848 100.003152
14: 99.786000 99.732768 99.839232
15: 94.854000 94.599496 95.108504
16: 53.082000 52.507122 53.656878
17: 2.592000 2.408960 2.775040
18: 0.000000 0.000000 0.000000
19: 0.000000 0.000000 0.000000
20: 0.000000 0.000000 0.000000

Path	Largest	Shortest	MIN	MAX	MEAN
1:	100.00	0.00	64.598	88.620	80.002
2:	0.00	0.23	48.671	75.624	64.997
3:	0.00	17.38	43.492	64.938	57.984
4:	0.00	2.72	48.499	64.372	59.999
5:	0.00	79.66	41.916	61.295	54.998

Activity	MIN	MAX	MEAN
1:	3.790	17.735	12.997
2:	11.090	26.516	22.004
3:	31.521	49.297	45.000
4:	18.691	34.138	29.996
5:	14.746	29.120	25.000
6:	19.457	34.326	29.999
7:	33.940	49.090	44.987
8:	48.499	64.372	59.999

- ***A380-800[75% de Ocupación]***

INPUTS FROM: T-CPA_inputs_first.txt
50000 8 20 5.000000 5 5588

INPUTS FROM: T-CPA_inputs_second.txt
w 7.819000 11.693500
w 13.169700 18.720600
w 33.905899 45.740398
w 22.380800 30.734301
w 18.539101 25.730499
w 22.378599 30.734400
w 33.905899 45.740398
w 45.438900 60.743099

3
3
2
1
2

INPUTS FROM: T-CPA_inputs_third.txt

1: 1 3 2
2: 1 4 2
3: 1 7
4: 8
5: 5 6

SIMULATION OUTPUTS:
1 74.022917 2.881844 73.989718 74.056115

1: 100.000000 100.000000 100.000000
2: 100.000000 100.000000 100.000000
3: 100.000000 100.000000 100.000000
4: 100.000000 100.000000 100.000000
5: 100.000000 100.000000 100.000000
6: 100.000000 100.000000 100.000000
7: 100.000000 100.000000 100.000000
8: 100.000000 100.000000 100.000000
9: 100.000000 100.000000 100.000000
10: 100.000000 100.000000 100.000000
11: 100.000000 100.000000 100.000000
12: 99.998000 99.992848 100.003152
13: 99.616000 99.544754 99.687246
14: 90.962000 90.631708 91.292292
15: 39.282000 38.719416 39.844584
16: 0.794000 0.691762 0.896238
17: 0.000000 0.000000 0.000000
18: 0.000000 0.000000 0.000000
19: 0.000000 0.000000 0.000000
20: 0.000000 0.000000 0.000000

Path	Largest	Shortest	MIN	MAX	MEAN
1:	99.98	0.00	58.098	83.469	74.023
2:	0.00	6.32	41.680	68.547	59.010
3:	0.00	33.54	41.082	63.159	56.024
4:	0.02	1.60	48.573	64.023	59.989
5:	0.00	58.54	40.296	61.752	54.999

Activity	MIN	MAX	MEAN
1:	3.371	15.836	11.016
2:	9.135	22.419	18.004
3:	33.151	49.312	45.002
4:	19.381	34.277	29.989
5:	15.161	29.164	24.998
6:	19.677	34.267	30.001
7:	31.951	49.183	45.009
8:	48.573	64.023	59.989

- ***B747-8F***

INPUTS FROM: T-CPA_inputs_first.txt
50000 7 20 5.000000 5 2917

INPUTS FROM: T-CPA_inputs_second.txt
w 34.683102 46.740398
w 34.674801 46.740601
w 14.702700 20.724300
w 14.702700 20.724300
w 53.128899 70.744202
w 33.886799 45.740799
w 22.126101 30.742300

2
2
1
1
1

INPUTS FROM: T-CPA_inputs_third.txt

1: 1 2
2: 3 4
3: 5
4: 6
5: 7

SIMULATION OUTPUTS:
1 91.999519 2.366011 91.972263 92.026775

1: 100.000000 100.000000 100.000000
2: 100.000000 100.000000 100.000000
3: 100.000000 100.000000 100.000000
4: 100.000000 100.000000 100.000000
5: 100.000000 100.000000 100.000000
6: 100.000000 100.000000 100.000000
7: 100.000000 100.000000 100.000000
8: 100.000000 100.000000 100.000000
9: 100.000000 100.000000 100.000000
10: 100.000000 100.000000 100.000000
11: 100.000000 100.000000 100.000000
12: 100.000000 100.000000 100.000000
13: 100.000000 100.000000 100.000000
14: 100.000000 100.000000 100.000000
15: 100.000000 100.000000 100.000000
16: 99.992000 99.981697 100.002303
17: 99.142000 99.035756 99.248244
18: 81.486000 81.038571 81.933429
19: 7.876000 7.565707 8.186293
20: 0.000000 0.000000 0.000000

Path	Largest	Shortest	MIN	MAX	MEAN
1:	100.00	0.00	76.570	98.553	92.000
2:	0.00	0.10	26.749	47.807	39.991
3:	0.00	0.00	57.638	74.003	70.002
4:	0.00	0.00	31.100	49.406	45.006
5:	0.00	99.90	18.245	34.395	29.989

Activity	MIN	MAX	MEAN
1:	33.834	49.987	46.002
2:	31.083	50.044	45.997
3:	10.818	24.151	19.999
4:	10.120	24.604	19.992
5:	57.638	74.003	70.002
6:	31.100	49.406	45.006
7:	18.245	34.395	29.989

- ***B757-200PF***

INPUTS FROM: T-CPA_inputs_first.txt
50000 6 10 5.000000 4 32572

INPUTS FROM: T-CPA_inputs_second.txt
w 18.539101 25.730499
w 18.539101 25.730499
w 14.702800 20.724300
w 14.702600 20.724300
w 10.872700 15.712800
w 3.302100 5.573900

1
1
2
2

INPUTS FROM: T-CPA_inputs_third.txt

1: 5
2: 6
3: 3 4
4: 1 2

SIMULATION OUTPUTS:
1 50.015721 2.328307 49.988899 50.042543

1: 100.000000 100.000000 100.000000
2: 100.000000 100.000000 100.000000
3: 100.000000 100.000000 100.000000
4: 100.000000 100.000000 100.000000
5: 100.000000 100.000000 100.000000
6: 100.000000 100.000000 100.000000
7: 100.000000 100.000000 100.000000
8: 99.970000 99.950051 99.989949
9: 97.142000 96.950059 97.333941
10: 54.250000 53.676112 54.823888

Path	Largest	Shortest	MIN	MAX	MEAN
1:	0.00	0.02	5.842	19.321	14.995
2:	0.00	99.98	0.165	11.488	4.991
3:	0.24	0.00	25.701	46.694	40.007
4:	99.76	0.00	37.311	56.790	50.013

Activity	MIN	MAX	MEAN
1:	12.760	29.389	25.002
2:	14.844	29.445	25.011
3:	10.408	24.451	20.002
4:	9.683	24.461	20.006
5:	5.842	19.321	14.995
6:	0.165	11.488	4.991