Proceedings of the 2007 IEEE
Intelligent Transportation Systems Conference
Seattle, WA, USA, Sept. 30 - Oct. 3, 2007

Simulation-based Input Loading Condition Optimisation of Airport Baggage Handling Systems

Vu T. Le [1], Dr Doug Creighton[2], Prof Saeid Nahavandi[3], Senior Member, IEEE

Abstract— **Scheduling check-in station operations are a challenging problem within airport systems. Prior to determining check-in resource schedules, an important step is to estimate the Baggage Handling System (BHS) operating capacity under non-stationary conditions. This ensures that check-in stations are not overloaded with bags, which would adversely affect the system and cause cascade stops and blockages. Cascading blockages can potentially lead to a poor level of service and in worst scenario a customer may depart without their bags. This paper presents an empirical study of a multiobjective problem within a BHS system. The goal is to estimate near optimal input operating conditions, such that no blockages occurs at check-in stations, while minimising the baggage travel time and maximising the throughput performance measures. We provide a practical hybrid simulation and binary search technique to determine a near optimal input throughput operating condition. The algorithm generates capacity constraint information that may be used by a scheduler to plan check-in operations based on flight arrival schedules.**

***Index Terms*— BHS, network analysis, conveyor system, merge, optimisation, operating policies.**

I. Introduction

Baggage handling systems are conveyor-based networks that transfer baggage (bags) from service check-in and transfer stations to output piers (laterals or make-up loops). The baggage associated with a particular flight is assigned to a specific pier or set of piers for an allocated time window. After a customer checks in at the service station, their bags are transferred to a delivery conveyor and then into the baggage handling system. Raised levels of security now require 100% of bags to pass a security screening procedure, within this network, before dispatch to their plane.

A BHS is a rapid material transfer medium designed such that the probability of a system resource failure is relatively low. This is because the system is fragile to any type of in-system operation or resource failure. Any of these failures would create a large impact to the system performance, as bags gets miss tracked, pile up in a certain area in the network, backlog it self and clogging the check-in stations which then raising problems from area to area and in the worse ever scenario it would overhauling the current airport operational activities. In these situations, one problem will give raise to another dilemma, causing the whole system to collapse.

In this paper we present a deterministic binary-based heuristic algorithm to solve a real world multi-objective goal optimisation problem. The aim is to determine the near optimal input flow rates, in order not to block check-in stations, while maximising the output throughput and minimises baggage travel time for BHS systems. This paper is organised as follows: Section II we review previous work; Section III define the BHS environment; Section IV we describe the algorithm; Section V we estimate the solution bound; Section VI experimental results and discussion were given; and Section VII is the concluding remark to our work.

II. Related Literature

An operational impact evaluation of airport passenger security systems was investigated by Pendergraft, Robertson and Shrader [1]. Simulation was successfully used to provide support and aid decision making with regard to resource requirements under different input loading, operating and staffing conditions and process layouts in order to increase service level. In our work the performance of the BHS is the source that provides capacity constraint for resource requirement planning. Simulation-based multi-objective optimisation was applied to a scheduling problem in the postal service industry by Persson et al. [2]. They modelled a mail sorting process, using the ARENA software package, and applied genetic algorithms to find an optimal mail schedule. The efficiency of the process was improved by performing rough estimation of the solutions before evaluating them on the time-consuming simulation. We employ a similar strategy in our algorithm, making a rough estimate of the upper and lower bounds of input flow rates to simulate, to reduce the number of unnecessary simulation runs.

In order to prevent or recover from system malfunctions and stoppages resulting from system overload, Lim and Jeong [3] applied simulation to evaluate difference loading and operating conditions on the Automatic Raw Material Inspection System (ARMIS). This provides the information so that good control logics can then be implements. This is similar to our work, where different loading conditions were applied to determine the system deliverable capacity, while trying to prevent potential developed bottleneck at check-ins stations without changing the control logics.

[1]V. Le is with the Intelligent Systems Research Lab, Deakin University. Geelong, Australia (vtl@deakin.edu.au).
[2] Dr. D. Creighton is with the Intelligent Systems Research Lab, Deakin University. Geelong, Australia (dougc@deakin.edu.au)
[3] Prof. S. Nahavandi is with the Intelligent Systems Research Lab, Deakin University, Geelong, Australia (nahavand@deakin.edu.au).

1-4244-1396-6/07/$25.00 ©2007 IEEE.

The combination of simulation, data mining and knowledge-based techniques is an emerging methodology and has been addressed in Painter et al. [4]. The method was employed to determine short term and long term impacts of aircraft engine maintenance decisions, based on life-cycle cost and operational availability in the Air Force. A clustering data mining technique was used to analyse simulation output to extract the subsystem interaction behaviour and the life-cycle. It was found that the method circumvents the risks and short comings of parametric model-based approaches and minimises dependency on historic maintenance.

Binary search [5] is a well known and most effective way to search sorted data in collections, with a $O(\log n)$ solving time. Over the years it has been further improved and has evolved to different search forms. Onak and Parys [6] extended the binary search technique to search trees and forest-like structures, with partial order sets, resulting in an upper bound computation time of $O(n^3)$. This was an improvement over the algorithm developed by Ben-Asher et al. [7], which had a solving time of $O(n^4\log^3 n)$. Chan et al. [8] derived an insertion sort, based on repeatedly performing binary search in an unordered array, having a running time of $O(n^2\log n)$. Some other evolved forms of binary search include the Fibonacci search [9, 10] and the interpolation search [11, 12]. Both of these algorithms generally perform faster than the traditional binary search. The Fibonacci search runs at $O(\log n)$, while the interpolation search has an $O(\log(\log n))$ searching time.

Our work applies *divide and conquer* binary search on each input node for a series of deterministic ordered sets of bag loading, to obtain a near optimum input operating condition. The next section provides a description to the problem investigated in this paper.

III. BHS Environment

The baggage handling system operates in a way such that when a bag enters the system, after check-in or bag transfer from other planes, it goes through the x-ray security screening machine to ensure that no dangerous goods or threats are loaded onto a plane. If something suspicious is identified, the bag will be transferred to the next level of security screening. Eventually, the bag will travel to a manual handling department if all levels of screening are unable to pass the bag or a threat is positively identified. If the bag is passed during screening, it then travels through many merges onto the main line, where it is identified by a scanning machine, called an Automatic Tag Reader (ATR). The controller allocates the time windows for the bag at the ATR, so that when a bag arrives at an output pier within a determinate time window, it gets pushed by the pusher to the expected output. If the bag arrives earlier or later at the output it would be identified as *unknown* and re-looped back into the system where it would be rescanned and reallocated to a new time window.

Baggage handling systems are generally rigid in structural design. However, when in operation, any unplanned system loading situation would easily cause the system to become a bottleneck. Resource planning and scheduling of check-in stations that didn't consider the BHS capacity constraint would easily overload a particular conveyor section within the system. This overloaded problem would cause bags to accumulate over time, congesting adjacent conveyor lines and causing check-in input queue to grow, which greatly affecting the level of service.

In seeing that the input loading rate is an important measurement to the BHS system, this paper performs optimisation study on the BHS simulation model illustrated in Figure 1. The aim is to determine the input operating parameters that would minimises the overall baggage travel time and optimises the output throughput and no blockage should occur to the check-in stations.

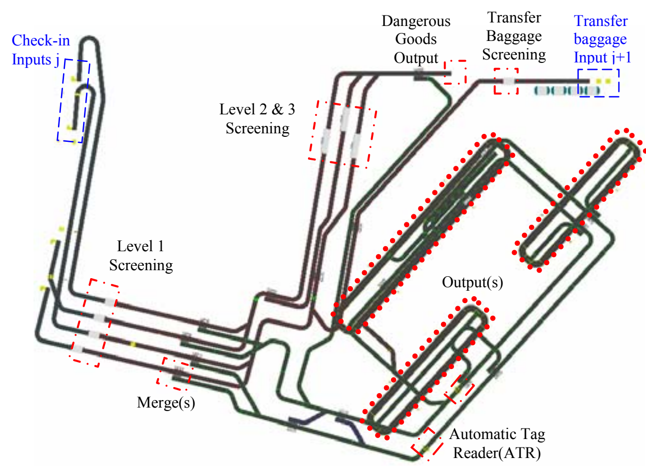

Figure 1: An example of a small size airport BHS.

The BHS shown in Figure 1 is an example of a smaller size system. It could be seen that, the system contains a number of merges that transfer bags from one conveyor to the next. The complexity of the merging rules could significantly impact system performance. In order to reduce the model complexity, this paper assumed that merge rule follows a standard FIFO configuration, which operate similar to the actual system. To further simplify the problem, it is reduced to a simple black box with input and output characteristic as illustrated in Figure 2.

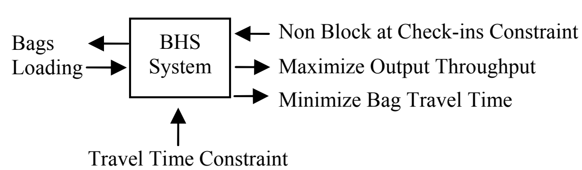

Figure 2: Schematic diagram of the simulation-based optimisation system.

IV. The Algorithm

The problem is derived such that there are a set of bags $\{b_i, b_{i+1}, b_{i+2} \ldots b_{i+n}\}$, where i is the bag index, that feed into each group of check-in and transfer input stations *j*. This occurs in such a way that the interarrival time T_{aj} is deterministic, $T_{aj} \in \{T_1, T_2 ... T_k, T_m\}$. T_1 is the minimum interarrival time, k is the interarrival time element index and T_m is the maximum interarrival time, where T_1 is the lower and T_m is the upper bound of the input flow rate for each check-in, $j \in \{1,2,3...N\}$ respectively.

The problem is solved by altering the input flow rate at each simulation run to each check-in input group. This means that there would be k^N number of simulation runs that needed to be solved. Since each simulation is running for T_s, $s \in \{1...H\}$, the total estimated simulation time and running time is $\sum_{s=1}^{H} T_s k^N$ or an $O(k^N)$ time problem type. Both the size of the problem and the chosen value of T_s will influence the actual simulation run time and optimisation solving time.

In order to reduce the problem complexity and solving time the binary search strategy that traverses all the hierarchical B-Tree branches of data structure, to search for a satisfactory solution within the simulation runs. The search criterion reduces the simulation run times to compute for a feasible region in *O*(NlogN) time. The algorithm to find the optimal feasible solution for the simulation has been given schematically summarised in Figure 3. The embedded binary search routine used to update run time variables, BinarySearchUpdate (BreakOn, IsBreak), is shown in Figure 4.

The algorithm defined in Figures 3 and 4 behaves such that there are N order sets of bags interarrival time $\{T_1, T_2, \ldots, T_k, T_m\}$, which are prepared at each of the input j. Initially each input is loaded with the lower bound of bags interarrival time T_1 and the simulation is run for a predetermined simulation time, T_s. If during this run time a check-in (j) becomes blocked or any check-in queue is greater than the soft queue limit constraint, the baggage interarrival time, T_{aj}, will reset on j according to Equation 1. The blocked check-in (j) is stamped and added to a current break list $L_{bo}(N)$. The simulation then waits for all the bags to drain out of the system with a set cool down time, T_c. The simulation is then re-run with this new loading condition.

$$T_{aj} = ½ (T_{kj} + T_{mj}) \qquad (1)$$

If an alternative check-in j is blocked on the next run or the subsequence runs, the binary search pointer moves to this currently block check-in, j, where the bag loading interarrival time will compute using Equation 1, while the new break on input j is added to the list $L_{bo}(N)$. If the previous check-in j is blocked, its stamped information will not add to the list $L_{bo}(N)$ as it is already in the list.

In the situation where no blockage has occurred, the baggage output throughput and travel time get stamped. The weighted cost is computed on this stamped data and is compared to the previous results, where the optimal condition is stored. If no obstructions occurred in the last two runs, it assumed that no blockages will occur for the current loading conditions, and the last blockage item N is then removed from the list $L_{bo}(N)$. The current pointer will be set to the previous element j, where binary search will resume on this current check-ins input j element.

The simulation is continued until there are no further improvements on the results for three consecutive runs and there is one item left in the list $L_{bo}(N)$. If there are more items left in the blockage list $L_{bo}(N)$ and the result doesn't alter in 3 runs, the search algorithm is assumed to be stuck on a local optimal solution. In order to escape from this local optimality the algorithm in Figure 4 picks a random check-in and runs with a different baggage interarrival time on this selected check-ins input. At the moment, this local optimum solution escaping routine just randomly pick a previously blocked check-in input and alter it on the next simulation run. It currently doesn't take priority rules into the selection criteria. After attaining the results, the loading parameter get resets and the simulation running period is increased.

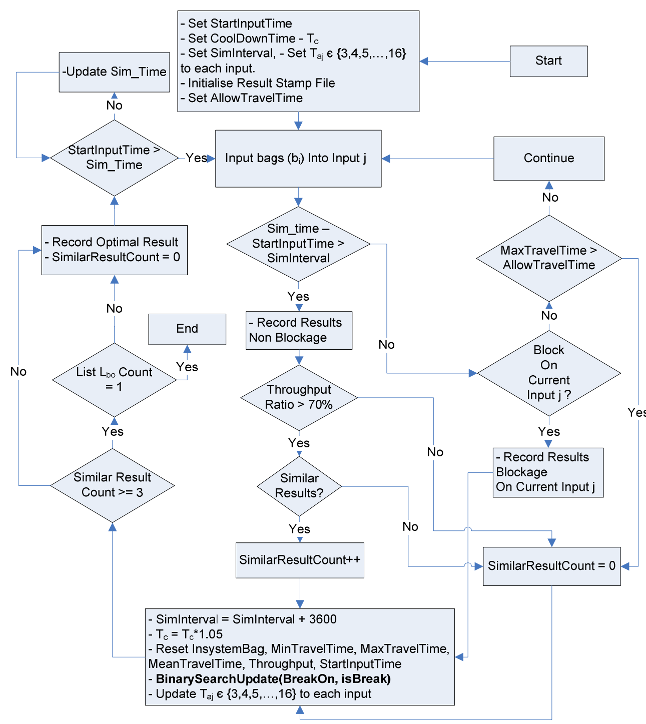

Figure 3: The embedded simulation optimisation algorithm.

In this work the simulation programming goal is that all bags have to exit the system within the travel time constraint and the throughput ratio (output/input) should be at least 70% within the simulation run interval for the current optimise input interarrival time loading solution. The 70% constraint is a safety factor which allows less than 30% of bags to accumulate within the system for a determine duration monitored interval. The 30% build up of bags downstream may raise a problem at the later stage, however, if the bags travel time are still within the capacity limit on the current simulation time run interval then the results are still valid and can possibly be a satisfactory solution.

```
BinarySearchUpdate (string curElement, bool IsBreak)
Begin
  E = GET_ELEMENT(curElement)
  Switch(IsBreak)
 Case True:
    If(curElement NotIn BlockageList_Lbo(N))
          UpdateBlockageList_ Lbo(curElement)
    EndIf
    Remainder = (E->InterATime + E->MaxInterTime) Mod 2
    If (Remainder == 0)
       E->IntATime = (E->IntATime + E->MaxIntTime)/2
     Else
       E->IntATime = (E->IntATime + E->MaxIntTime - 1)/2
    EndIf
  Case False:
    If (maxTravelTime > TravelTimeConstraint)
          If(stuckOnLocalSolution)
             E =  SelectRandomCheckinFromList_Lbo(N)
          EndIf
          E->doubleMinInterTime = E->doubleMidInterTime
          E->doubleMaxInterTime = intMaxInterArrivalTime
    Else
          E->doubleMaxInterTime = E->doubleMidInterTime
    EndIf
    Remainder = (E->InterATime + E->MaxInterTime) Mod 2
    If (Remainder == 0)
       E->IntATime = (E->IntATime + E->MaxIntTime)/2
     Else
       E->IntATime = (E->IntATime + E->MaxIntTime + 1)/2
    EndIf
  End Switch
End
```

Figure 4: Variable updating binary search technique.

In order to reduce the simulation running time and search space, determining of the appropriate solution bound, T_1 and the T_m, is a crucial initial step in order to avoid unneeded simulation search time in large model. The next section attempt to estimate this inter-arrival time bound.

V. Simulation Bounding Constraints

This section attempt to estimate the bag inter-arrival time solution bound T_1 and T_m to optimise simulation running time and preventing unnecessary processing time.

The first bottleneck affecting the lower bound T_1 value is the delivery conveyor at the check-ins. The inter-arrival of bags into the system should not be greater than the input conveyor takeaway capacity speed. That is, if the conveyor takeaway rate is $\dot{C}$ (units of bags per hour), then $T_1 \geq \dot{C}/3600$, so that blockage will not occur.

Blockages can also be caused by screening machines, if the processing rate is lower than the input feeding rate. Hence, the optimal design would be having the x-ray screening machine operating at equal to or greater than the conveyor delivery capacity. This ensures that the chance for a bottleneck to develop is minimised. In the situation where screening machine processing rate is lower than the input delivery rate, ensures that bags injection rate are less than the screening machine processing rate. This is to prevent any blockages building up at these sections that would deteriorate the system performance, whilst lowering the level of service. This would mean that the lower bound, T_1, should be chosen to be greater than the mean interarrival of bags to further reducing the solution space and unnecessary simulation time.

The upper bound mean interarrival time T_m was based on interarrival distributions generated from actual data collected at airport check-in and take away conveyors. The frequency of bags arriving into the system against interarrival time on a group of check-ins for flights has been plotted in Figure 5. This figure illustrates that the peak bag interarrival frequency is in between 6 and 7 seconds, at the mode values of data set. In this situation the mean bag interarrival time value lies somewhere to the right of the mode frequency. Hence the mode is more important to see that if the conveyor delivery capacity is broken, so that the previously chosen T_1 value could be overwritten.

The mean interarrival time value can be used as an initial estimate to the upper bound T_m value. As long as a sensible value is chosen and the solution space constraint doesn't overload the system and $T_m > T_1$, then T_m is a soft constraint that could be arbitrary chosen.

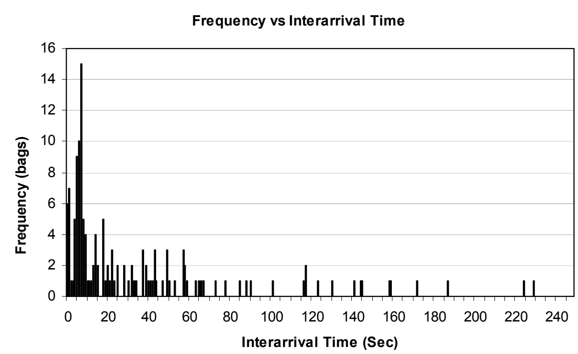

Figure 5: Bags input frequency (1 second bin interval).

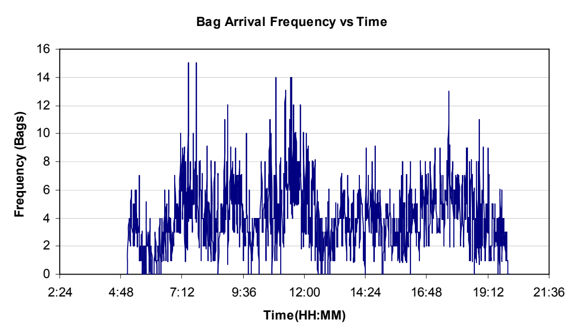

Figure 6: Daily bag input frequency profile.

The baggage input frequency (in one minute buckets) to one of the check-in groups for one day is given in Figure 6. This is the condition when check-in stations are serving for one to more flight schedules, at different time periods throughout the day. In the early morning there would be one to two flights on that check-in group, while during the afternoon peak a higher number of flights would be serviced. This graph shows the combination of multiple sets of periodical data, similar to Figure 5.

VI. Experiment and Results

In this section, we present the investigation outcomes of the binary algorithm, which was applied to the BHS system in Figure 1. The respective input loading interarrival time parameters have been given in Table I. The transfer input has been set to a constant of 10 seconds interarrival. This is because we would like to study the effect caused by check-in input loading variation independent of the transfer bags input. Transfer baggage arrives in batches of 20 in the simulation, so a faster interarrival time at the transfer input would significantly affect the travel time of baggage merging from the Level 2 and Level 3 screening process.

TABLE I
INPUT LOADING CONDITION

Input Type	Input (j)	T_{aj} (secs)
Check-in	1	{3,4,..,16}
Check-in	2	{3,4,..,16}
Check-in	3	{3,4,..,16}
Check-in	4	{3,4,..,16}
Transfer Input	5	10

In order evaluate the performance of the binary search algorithm, six sets of simulation runs have performed. The mean optimal results to each of these simulation runs for a non constraint travel time have been summarised in Table II. When 30 minute constraint is applied to the bag travel time, the results have been given in Table III. In both tables, it could be seen that the optimal solution for the simulation model having the bag interarrival time of four seconds, which is equivalent to 900 bags per hour, running for five hours straight without blockage develop at the check-ins.

TABLE II
NON BAG TRAVEL TIME CONSTRAINT SIMULATION RESULTS

	Run Results					
	$T_{aj,min}$					
Input	**1**	**2**	**3**	**4**	**5**	**6**
1	4	4	4	4	4	4
2	3	3	3	4	4	4
3	4	4	4	4	4	4
4	3	4	4	4	4	4
5	10	10	10	10	10	10
Interval Run Time (Secs)	1800	3600	7200	10800	14400	18000
Cooling Time	1800	1889	1983	2082	2186	2295
Max Travel Time (Secs)	1045	1048	1108	1567	1112	1895
Min Travel Time (Secs)	212	212	212	212	212	212
Mean Travel Time(Secs)	386	383	388	384	383	380
Output Throughput (bags/hr)	3352	3742	3989	3792	3833	3858
Throughput Ratio (%)	73.51	87.84	93.64	95.58	96.79	97.42
Iterations	11	13	13	17	17	19

By comparing the results for constrained and unconstrained travel time in Table II to Table III, it can be seen that visually there are no significant differences between the results. The near optimal bag interarrival time, maximum, minimum, mean travel time and throughput are comparable between the two. This was due to the travel time constraint being greater than the actual travel time for the majority of the bags. This means that the number of iteration to obtain a good solution is approximately similar between the two, as shown in Figures 1 and 2 respectively. Any differences between the results were caused by the stochastic processing screening rate security check for dangerous goods and potential threats. A percentage of bags *miss screen*, on the first pass through the x-ray machine, and are required to be rescreened. These backs must travel to the next level before being rerouted back into the main loop. Bags may not be scanned correctly at Automatic Tag Readers and must be routed to a manual scanning station for identification before going back into the main system. These unexpected situations affect the magnitude of bags travel time and throughput. These events also give rise to the build up of bags before local bottlenecks causing alternative check-in stations to blocks and affect the number of iteration runs to obtain a good set of solutions.

TABLE III
30 MIN BAG TRAVEL TIME CONSTRAINT SIMULATION RESULTS

	Run Results					
	$T_{aj,min}$					
Input	**1**	**2**	**3**	**4**	**5**	**6**
1	4	4	3	4	4	4
2	3	3	4	3	4	4
3	4	4	4	4	4	4
4	3	4	4	4	4	4
5	10	10	10	10	10	10
Interval Run Time (Secs)	1800	3600	7200	10800	14400	18000
Cooling Time	1800	1889	1983	2082	2186	2295
Max Travel Time (Secs)	1045	1048	1106	1285	1072	1189
Min Travel Time (Secs)	212	212	212	212	212	212
Mean Travel Time (Secs)	386	384	381	396	377	379
Output Throughput (bags/hr)	3353	3743	4005	4075	3835	3855
Throughput Ratio	73.53	87.86	94.01	95.57	96.84	97.25
Iterations	11	13	14	14	18	17

In situation when the travel time constraint is reduced so that no bag should be in the system longer than 16 minutes. The reason 16 minutes was chosen here instead of 15 minutes was due to no feasible solution been found at the 15 bounding constraint. Any feasible solution found for the 15 minute constraint on this system is pure luck that contributed by lower missed screening and lower false tag reading rate, which causes bags to exit the system earlier. The optimised results obtained on the 16 minute travel time constraint are given in Table IV. The results have shown that bag loading interval has to increased, in order to meet the travel time requirement. As a consequence of this, the throughput is affected and reduced and the number of iteration required increases. The output measures on each sets of simulation subinterval running time for the travel time constraints equals to 16 minutes and 30 minute constraints are given in Figure 7.

TABLE IV
16 MIN BAG TRAVEL TIME CONSTRAINT SIMULATION RESULTS

	Run Results					
	$T_{aj,min}$					
Input	**1**	**2**	**3**	**4**	**5**	**6**
1	13	16	12	16	15	16
2	10	10	16	13	16	16
3	16	16	10	16	16	16
4	3	10	12	10	10	16
5	10	10	10	10	10	10
Interval Run Time (Secs)	1800	3600	7200	10800	14400	18000
Cooling Time	1800	1889	1983	2082	2186	2295
Max Travel Time (Secs)	896	899	886	923	933	933
Min Travel Time (Secs)	212	212	212	212	212	212
Mean Travel Time (Secs)	379	368	327	331	325	299
Output Throughput (bags/hr)	1833	1353	1450	1313	1287	1198
Throughput Ratio	75.68	88.43	93.85	90.74	91.28	95.08
Iterations	10	16	14	17	15	28

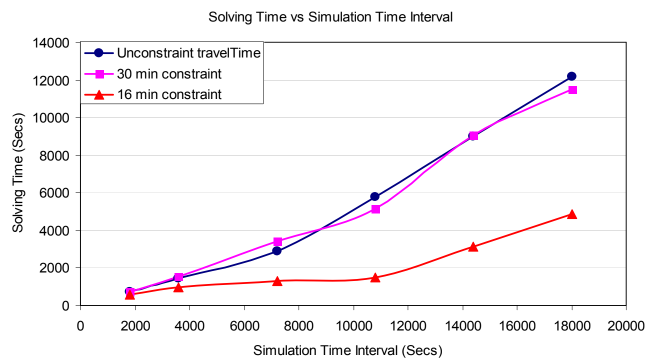

Figure 7: Comparison between constraint and non constraint travel time.

The plotted results in Figure 7 for the solution time in the unconstrained and 30 minutes travel time constrained case show that both have comparable trend running time intervals. The solving time took approximately half of the duration compare to the simulation time at the lower data values. As the simulation time interval increases the solving time rapidly catches up to the simulation interval time. In the 16 minute scenarios, the rate of change in solving time is relatively low. This situation has been facilitated by the algorithm such that as a bag leave the system, it travel time instantly been evaluate against the constraint. If a bag travel time is greater than the constraint time, the simulation restart and iterate through with new sets of loading conditions. This gives the effect that the solving time being lower. Although the solving time is lower, the actual number of iterations are higher, compared to the 30 minutes constraint and unconstraint experiment results.

VII. CONCLUSION

In this paper we have developed a hybrid simulation and binary search technique on the hierarchy B-Tree data structure, to optimise the baggage loading condition at check-in and transfer input stations for an airport BHS. This minimised the chance of a cascade stop traverse upstream from a merge bottleneck. Such blockages would significantly impact the customer service levels of an airport and airline. The optimised operating policy aids the scheduler in generating check-in station operational schedules, by providing schedulers with the estimated baggage loading capacity constraints. Studies have shown that the integrated algorithm could obtain the results in a manageable time. When the bag travel time constraint is reduced, both the simulation time and the number of iterations required to be performed increases and the input and output throughput rates are reduced.

In this work the check-in input queuing constraint have been limited to one bag for each of the input. It could be an extension to monitor the effect from varying the input queue length and studying the affect from varying the input loading conditions. This would further increase the complexity scope of the underlying problem.

The heuristic algorithm is an ideal integration with simulation environment to solve a multi-objective problem, like the BHS system in our investigation. It could be employed in transportation to estimate loading conditions and road blockages due to downstream traffic.

Acknowledgement

This project was funded by an ARC Linkage Project under contract number LP0454009.

REFERENCES

[1] D. R. Pendergraft, C. V. Robertson and S. Shrader, "Simulation of an airport passenger security system", Proceeding of the 2004 Winter Simulation Conference. pp. 874-878, 2004.
[2] A. Persson, H. Grimm, Ng. Amos, T. Lezama, J. Ekberg, S. Falk and P. Stablum, "Simulation-based multi-objective optimization of a real-world scheduling problem", Proceedings of the 2006 Winter Simulation Conference. pp. 1757-1764, 2006.
[3] T. G. Lim and K. W. Jeong, "Evaluation of operation load in automatic raw material inspection system", In IEEE 2003. pp. 870-875, 2003.
[4] M. K. Painter, M. Erraguntla, G. L. Hogg Jr. and B. Beachkofski, "Using simulation, data mining, and knowledge discovery techniques for optimised aircraft engine fleet management", Proceeding of the 2006 Winter Simulation Conference. pp. 1253-1260, 2006.
[5] D. E. Knuth, "The art of computer programming: sorting and searching", Addison-Wesley, Massachusetts, second edition Vol. 3. pp. 412, 1998.
[6] K. Onak and P. Parys, "Generalization of binary search: searching in trees and forest-like partial orders", Warsaw University. Aug14. 2006.
[7] Y. Ben-Asher, E. Farchi and I. Newman, "Optimal search in trees", SIAM Journal on Computing. 28(6), pp. 2090-2102, 1999.
[8] T. Biedl, T. Chan, E. D. Demaine, R. Fleischer, M. Golin, J. A. King and I Munro, "Fun-sort-or the chaos of unordered binary search", on Science Direct Discrete Applied Mathematics. 144(3), 231-236, 2004.
[9] D. E. Ferguson, "Fibonaccian searching", Communication of the ACM. 3(12). pp. 648, 1960.
[10] B. Yildiz and E. Karaduman, "On Fibonacci search method with k-Lucas numbers", in Applied Mathematics and Computation, Elsevier Science Inc. 143(2-3), pp. 523-531, 2003.
[11] W. W. Peterson, "Addressing for random-access storage", IBM Journal of Research and Development, 1(2), pp.130-146, 1957.
[12] K. Mehlhorn and A. Tsakalidis, "Dynamic interpolation search", Journal of the ACM. 40(3), pp. 621-634, 1993.