META HEURISTICS

Spares Warehouse

Report of Genetic Algorithm Optimization

Brighid L Meredith

8/5/2014

Typical commercial aircraft are multimillion dollar investments which require frequent maintenance to guarantee airworthiness. Upon receiving such maintenance, if a part is found to be broken, then it must be repaired or replaced prior to the aircraft's flight. If the part is not on hand at the repair facility, then the Aircraft On Ground (AOG) results in significant lost revenues for each flight that it misses. On the other hand, if the repair facility contains a large inventory of spare parts, and if a significant portion of the inventory is not necessary, then the capital that has been spent on the inventory could have been better invested elsewhere, and the excessive inventory represents lost opportunity

Introduction:

Commercial jetliners turned the sky into a highway. Few if any travel alternatives exist that would satisfy today's business' needs. But while jetliners made traveling into a convenience, their continued operation requires logistics, dedicated personnel, and a lot of capital.

Aircraft are highly complicated pieces of machinery; Boeing's 737 Next Generation has over 600,000 parts. A portion of these parts are flight critical, meaning that the aircraft cannot fly, either by a technical or a legal restriction, until the part is restored to a serviceable state.

When an Aircraft is grounded due to a maintenance requirement, the term Aircraft on Ground (AOG) and when a shop is alerted to an AOG, they work as diligently and as quickly as possible to put the bird back in the sky. Aircraft are multimillion dollar investments. The 737 as an example, is listed at prices up to $87 million dollars. An AOG might cost thousands of dollars a day in lost revenue and even more in goodwill.

When a flight critical part breaks and if a long lead time exists for that part, the AOG can last for many days. This is an unfortunate place for the airline to be, with the cost being estimated at above $7000 per day. The best way around this problem is to keep a reserve of parts on hand, for those 'just in case' moments. This is a practical solution called spares.

But spares are inventory. They weigh down a company's books, depress profits, and lead to missed opportunities. And aircraft parts are very expensive. To give an example to the reader; a simple cabinet or closet, similar to that available at Ikea, when placed on an aircraft, would have a cost comparable to a humbly sized house.

The penalty of the inventory's cost can be calculated based on the operator's own internal interest rate. This is the payback required on an investment to justify the risk. Many companies or departments use an interest rate on the order of 10%

Between the penalty of inventory and the penalty of AOGs, the spares department is put in a difficult position; they only want the parts when required. This is a typical knapsack problem, for the spares department is limited on what it can have and not every item will prove worthwhile to the maintenance department. Operators cannot forecast maintenance to any great degree of certainty, and the logistics involved are messy.

If there were only one kind of part, the logistic would be easy, but considering the large number of parts used on an aircraft and their unique parameters, the problem becomes difficult. Parameters such as Mean Time between Failure (MBTF), cost, reparability, lead time, size, storage requirements, and shelf life all vary from part to part.

A few parts are Line Replaceable Units (LRUs). These are large components that can be removed intact from an aircraft, to be serviced in the maintenance shop. Many of these LRUs contain both Detail Elements (DEs) along with Sub-Assemblies (SAs). The Detail Elements cannot be broken down any further, nor can they be repaired. Detail Elements are such as resistors, blank cards, nuts, and bolts. Sub-

Assemblies are often made up of many Discrete Elements and occasionally contain their own Sub-Assemblies as well. These SAs may be reparable, depending on available documentation and feasibility. If the part isn't reparable, then the entire SA would need to be replaced when a single DE fails.

The matter of logistics is not only dependent upon the overall overhead, but also upon the individual component selection. If a shop stocks DEs, then SAs and LRUs might be repaired; but if the SAs or LRUs are not reparable, then they need to be stocked. The Spares Warehouse needs to stock the right combination and quantity of parts that are most likely to be required. As many combinations exist at different levels of cost, this problem is ideal for a meta-heuristic.

Data:

One of the difficulties of these logistics is to determine an appropriate format and source of data to use. Boeing naturally keeps track of this data, but it is proprietary, and it was not ideal for this project. To circumvent this difficulty, the data was artificially constructed based upon a reasonable distribution of values, for both properties and structure of the parts and their breakdowns. These values were set by years of experience in working with similar data in a parallel field.

To generate the data, a top down approach was utilized for structure, and a bottom up approach was used for cost and MBTFs. The first thing that was generated was a list of LRUs that may occur on the aircraft. Due to processing constraints, only ten different types of LRUs were considered for this process. Each of these LRUs is provided a unique part number, and each is also given a quantity per aircraft. The same process was followed for one hundred SAs and one thousand DEs.

With the lists of possible LRUs, SAs, and DEs to choose from, the breakdown of each component was generated. To each LRU, a list of SAs and DEs were assigned at random to form its first level of composition. The number of SAs assigned ranged from zero to fifteen, and the number of DEs assigned ranged from one to a hundred. For each SA, a range of SAs and DEs were assigned. There was a ten percent chance for a SA to contain another level of breakdown besides the DEs. If the SA contained another SA, then it contained between one and four additional SAs. The SAs were assigned DEs numbering between one and fifty. The quantity of each type of DE and SA assigned was random, ranging from a minimum of one to a maximum of five or six.

It would be prudent to highlight the various tiers of data that were found. The simplest data was composed of a single LRU without any SAs assigned, and only a handful of DEs. The most complicated data, a single LRU would contain multiple SAs, and these SAs could also contain additional levels of Sub-Assemblies. While increasingly complex structures are less common, they do occur on an aircraft, and the data captured this complexity in multiple places where there were tiers of three SAs and higher. Appendix A contains examples data for the breakdowns of both LRUs and SAs.

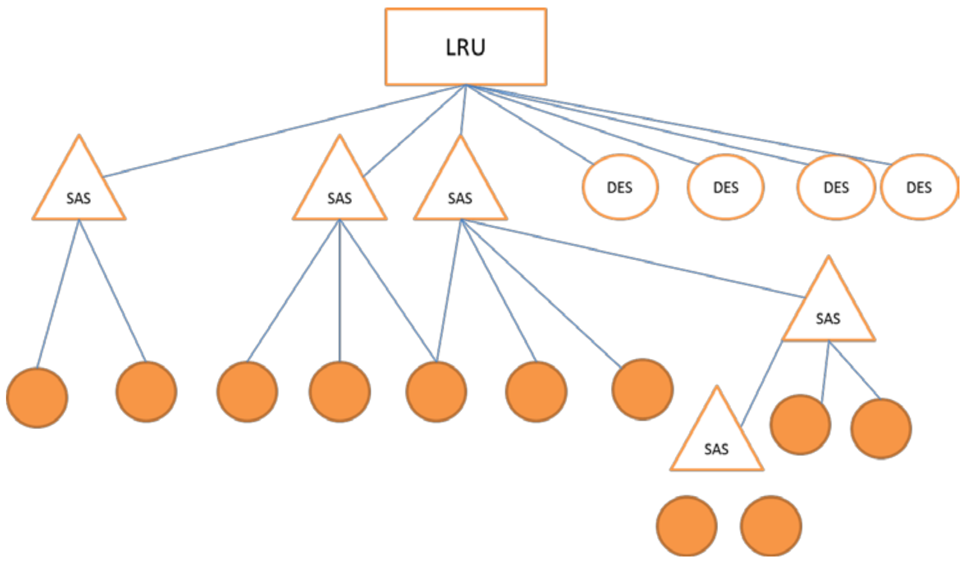

Figure 1: Example of a Three Tiered System

With the structure of the parts determined, only the data remained, and that included the MTBF, the Cost, the Lead Time when ordered, and if the component was reparable or not. To begin with, the DEs were assigned random MTBF based on a Gaussian distribution with a mean of five years and a standard deviation of three hundred days. The DEs were assigned a cost according to a random Gaussian distribution, with a mean of one hundred dollars, and a standard deviation of twenty five dollars. All DEs are considered non-reparable and the lead times were based upon a Gaussian curve with a mean set at fifteen days and a standard deviation of eight. The reparability for both SAs and LRUs was determined by a Gaussian curve centered on sixth, with a standard deviation of about one eighth. A random value was compared with the Gaussian value, and if the random value was greater, than the component was considered non-reparable. Lead Times were calculated in the same manner for the SAs and LRUs as they were for the DEs.

The MBTFs for SAs and LRUs were ignored, as only the DEs were considered as ‘breakable’. If a SAs was not reparable, then even if only a single DEs were destroyed, the entire unit would still need to be replaced.

The costs were calculated for the SAs based upon the sub components and DEs that made up the SA. The sum of all of the costs of the subcomponents was added, and then increased by forty percent. This was to duplicate the standard mark-up seen in industry for sub suppliers and manufacturers of aircraft parts. The same was performed for LRUs. The costs of the LRUs appeared reasonable, with both a good spread and an appropriate range. The cheapest LRU will not be less expensive than ten thousand, while the most expensive LRUs will cost upwards of hundreds of thousands of dollars.

The data was then encoded into Comma Separated Values (CSV) formatted files, and saved to the hard disk for later use.

Model:

The objective function will role play three years of quarterly maintenance. At the beginning of each quarter, each DEs has an opportunity to fail. Reliability was determined by an exponential fitness function that uses the MTBFs of each individual Discrete Element. When a part fails, the model will first determine if the part was in spares. If the part was found in spares, then the model subtracted the part from spares and moved forward. Otherwise, the part was put onto an ordering list.

After all the parts were tested, then the part on the ordering list with the longest lead time determined the AOG penalty. This longest lead time was multiplied by the daily lost profit of the grounded aircraft..

The objective function ran multiple times for each solution to determine a statistically relevant operation cost. See Appendix B for proof that the iterations provided a statistically relevant fitness. The operation cost is an indicator of solution fitness, where a lower cost indicates a better solution. Cost is composed of lost revenues from grounded aircraft and inventory levels. The average cost for the population served as a marker of fitness.

$$i \ldots for\ each\ discrete\ element\ used\ on\ aircraft$$

$$j \ldots for\ each\ Next\ Higher\ Assembly\ that\ a\ single\ discrete\ element\ is\ used\ on$$

$$l_{ij} \ldots lead\ time\ for\ part\ p_{ij}$$

$$r_{ij} \ldots\ reparability\ for\ part\ p_{ij}$$

$$c_s \ldots cost\ of\ part\ p_s\ in\ spares$$

$$r_{ij} \begin{cases} 10^9 \ldots non\text{-}reparable\ units \\ 1 \ldots\ldots\ldots for\ reparable\ units \end{cases}$$

$$Lead\ Time_{n\,q} = MAX\left\{\sum_{i=0} MIN\left(l_{ij} r_{ij}\right)\right\}$$

$$Spares\ Cost_{n\,q} = \sum_{s=0} c_s$$

$$Total\ Lead\ Time_n = \sum_{q=0}^{12} Lead\ Time_{n\,q}$$

$$Grounded\ Penalty = \frac{Aircraft\ Cost}{Aircraft\ Life}$$

$$Avreage\ Lead\ Time\ Penalty = \frac{Grounded\ Penalty}{N} \sum_{n=0}^{N\ Iterations} Total\ Lead\ Time_n$$

$$Inventory\ Penalty = \frac{Internal\ Interest\ Rate\ (12\%)}{4\ Quarters\ per\ Year}$$

$$Average\ Inventory\ Penalty = \frac{Inventory\ Penalty}{N} \sum_{n=0}^{N} \sum_{q=0}^{12} Spares\ Cost_{n\ q}$$

$$Total\ Penalty = Average\ Lead\ Time\ Penalty + Average\ Inventory\ Penalty$$

Algorithm:

A Genetic Algorithm (GA) was created to find an optimized solution, but the GA had several departures from tradition. Traditionally, a GA would include a mutation function where, upon reproduction, certain individuals may undergo a random aberration; this increases diversity and prevents premature convergence. Instead of a mutation, a random solution was generated. This method generated the initial population and introduced random solutions within each iteration. This widened the genetic stock within the population, and took the place of random mutations within the algorithm. Another departure from the traditional algorithm; this program analyzed the neighbors adjacent to every solution for every iteration. The net effect of these differences was that a level of diversification occurred before any solutions were selected for propagation.

The selection process used a roulette wheel methodology, where the best of the solutions, as compared to the population's average fitness, were given up to sixteen times the chance to reproduce as the weakest in the population. There was a range between the most fit and the worst, partitioned by the solutions overall fitness.

Vertical Recombination was used for reproduction, with four cut sets of randomly set lengths. The first and the third cut set were delivered by the first parent, and the second and third cut set were delivered by the second parent. Only a small fraction of the population was allowed to mate, and these couplings were forced to generate a large number of offspring. This tendency towards mono-diversity was offset by the diversification offered by the randomly generated solutions. Even if the random solutions are on average worse than the total population, by sheer chance alone, they will have an impact on the genetic makeup of the future generation.

There was one form of elitism utilized, and that was that the best solution was saved and inserted into the following generation. This prevented the best solution from ever regressing, but still allowed the remainder of the population to search for a better solution. If a better solution was not found over several iterations, then the population might begin to clone the best solution; however, this would be minimized by the random solution and the search for random neighbors, and eventually a better solution would be found.

The algorithm was terminated when the average health and fitness of the population was no longer improving after an uncertain number of iterations. When a following generation was worse than the preceding generation, than a tally was recorded, and if that tally exceeded a certain number, than the algorithm was terminated; however, it was an uncertain number of failed iterations that accomplished this, as a successful iteration would subtract from the tally, with a minimum set to zero.

```
Generate initial population size $N_{Pop}$:
For each solution in population
 Decide Random Length of Solution (number of spares)
 For each spare
   Pick random item from list of all equipment
While Solution is Still Improving
 Create $N_{Rand}$ new solutions at random, add to $X_{Pop}$
  For Each Solution in $X_{Pop}$
  Create $N_{fractals}$ new solutions at random,
  Each to be a neighbor of a previous solution
  Find the fitness of every individual in $X_{Pop}$
 Calculate the Average Fitness
 Calculate the Standard Deviation
 Record the most fit solution
 Selection
 For Each Individual in $X_{Pop}$
  Assign spots on roulette wheel based on fitness
  If solution is greater than average + Standard Deviation
  Solution receives one spot on wheel
  Else If solution is greater than average
  Solution receives two spots on wheel
  Else If solution is greater than average − Standard Deviation
  Solution receives four spots on wheel
  Else If solution is greater than average − 2 Standard Deviations
  Solution receives eight spots on wheel
  Else
  Solution receives sixteen spots on wheel
 Spin the Wheel …
 Randomly select Start of Selection
 While there are still Maters to Select $(mating_{fraction})$
  Select each mater based on equal distances determined by $mating_{fraction}$
  For Each Mater Selected
  Determine pairing of parents
  Determine order of two parents at random (1 or 2)
  For each Child this pairing has to create $(\frac{1}{mating_{fraction}})$
```

```
      $Determine\ three\ starting\ points\ at\ random, to\ be\ less\ than\ shortest\ parent$
      $New\ Child\ recieves\ genes\ from\ start\ to\ first\ cut\ from\ parent\ 1$
      $New\ Child\ receives\ genes\ from\ first\ cut\ to\ second\ cut\ from\ parent\ 2$
      $New\ Child\ receives\ genes\ from\ second\ cut\ to\ third\ cut\ from\ parent\ 1$
      $New\ Child\ receives\ genes\ from\ parent\ 2\ until\ parent\ 2\ is\ exhausted$
  $Replace\ X_{Pop}\ with\ new\ population$
  $Add\ Best\ Solution\ to\ X_{Pop}$
  $If\ Average\ Health - Last\ Average\ Health > -.01\ Standard\ Deviation$
  $Add\ to\ tally$
  $Else$
  $Subract\ from\ tally$
$Print\ Results$
```

Results:

A few general notes on the runs; the convergence depended upon several key parameters. The population size had to be greater than one hundred, or else no downward trend occurred. Also the number of random solutions introduced had a large affect upon convergence. If N_Rand was set too high, then the solution would not converge, but if it was set too low, then the average health would not improve. For the number of adjacent solutions, dictated by N_fractals, it was determined that if N_fractals was too large, then the algorithm became too greedy and took far too long to converge (at least with the available processors).

The number of iterations performed by the objective function affected the overall convergence. When N_Iterations was less than 10, the algorithm did not converge.

An ideal set of parameters was found that did lead to convergence, and these are listed in Table 1 below:

Parameters:	
N_Pop	200
N_Rand	20
N_fractals	2
N_Iterations	10
lt_penalty	7305.93607
c_penalty	0.03
Best Solution	$5,060,000
Total Saved	$430,000
	0.08498024
Run Time (hrs)	15
n (parts)	1110

Table 1: Run 1, the first to converge

This was the first run to successfully converge on an optimum solution. The best solution found had an overall cost that was just over five million, and it represented an eight and a half percent decrease in spares cost. The run time of this run was significant, caused in no small part to the large population size; using a standard laptop, it required fifteen hours to demonstrate convergence.

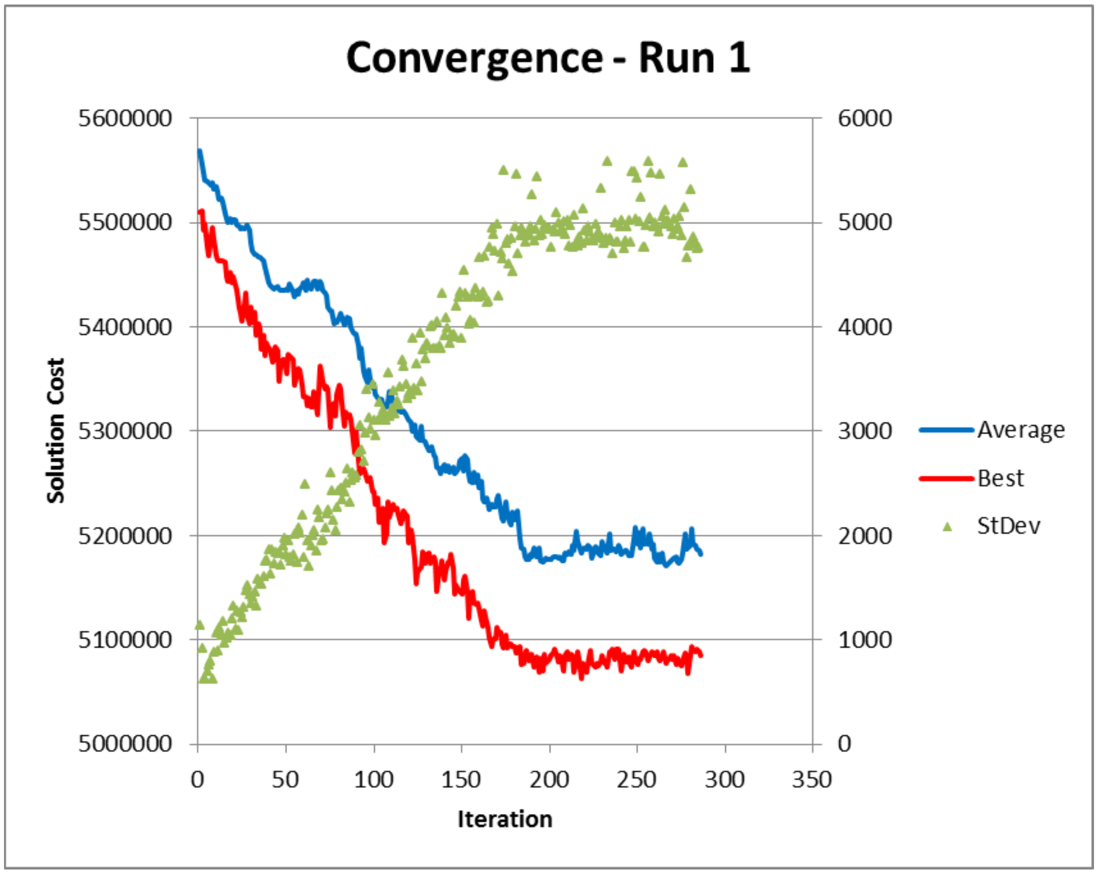

Figure 2: Run 1 Convergence

Discussion:

Spare Warehouses can pose a large and messy problem. They have many parts, and each part has unique parameters and requirements. But after several assumptions, the problem was simplified to a knapsack problem and machine learning strategies became feasible. Demonstration proved this algorithm can work on a limited part-set. But an actual fleet could have a part-set six hundred times larger.

As this algorithm already requires three minutes per iteration, or roughly fifteen hours to converge on a solution, there is reason to be concerned in regards to the scalability of this algorithm. While it was shown to work on a small subset of parts, it would require significant patience or computational power to expand this algorithm to a full fleet.

A potential avenue to consider is the use of cloud computing to offload the load to a larger processor. The author had some experience with Amazon Web Services (AWS), but due to an unfortunately large bill, the author has cancelled all business with AWS. If a corporation or business were to pursue this option, it is available, and it would likely prove to be a cheap alternative to dedicated servers.

Another point of concern; there is no known optimum for this solution, so there is no way to have complete confidence in the optimized solution returned. That being said, the best solution offered a ten percent improvement. Ten percent could make a big difference. Considering the limited testing and optimization, the improvement is outstanding and proves the algorithm’s applicability.

As a side note: A curious side effect was created by introducing a random set of solutions for each iteration. As the population’s average health improved and its cost decreased, the standard deviation increased. This occurred because the random solutions average at a worse health than the solutions being optimized and they would weigh heavily against both the standard deviation and the average health at large. This could pose problems when the standard deviation is used as part of the exit criteria for the algorithm. This is justified by the observation that the standard deviation climbs as the average improves, until both level out at convergence. At convergence, the average health does not approach the best health as would occur normally for a GA, but instead is weighted upwards by the influx of random solutions.

Conclusion:

The algorithm converged when the parameters were carefully selected, but could be improved further. Further optimization of the algorithm is possible, but a server environment is recommended. Given performance restrictions, the solution provided a ten percent improvement. While the algorithm used a data set limited by size, the algorithm demonstrated proof of concept for analyzing a larger data set for an actual aircraft fleet. With appropriate computational resources assigned, this algorithm could be modified to improve a standard spares warehouse. A ten percent savings is significant and would only be the start.

Appendix A

Sample of Data Used

LRU	Breakdown	Qty
100000		4
	300857	2
	300784	5
	300187	1
	300738	5
	300618	2
	300815	2
	300037	2
	300591	4
	300606	2
	300739	5
	200092	4
	200017	4
	200065	4
	200057	4
	200062	3
	200028	3
	200026	6

Table 2: LRU Breakdowns and Quantities

SAS	Breakdown	QTY
200012		2
	300832	5
	300643	3
	300055	6
	300869	3
	300431	6
	300180	1
	300138	5
	300869	7
	300050	4
	300975	7
	300316	2
	300668	6
	300681	5
	300570	3

	300963	2
	300923	7
	300627	7
	300105	5
	300731	2
	300247	1
	300850	6
	300392	7
	300572	5
	300824	6
	300260	4
	300132	2
	300016	6
	300953	6
	300667	5
	300730	6
	300805	7
	300660	4
	300782	2
	300308	6
	300526	1
	300129	6
	300278	4
	300725	5
	200091	3
	200003	5
	200066	1

Table 3: SAS Breakdowns and Quantities

Part	MTBF	COST	Reparable	Lead Time
300000	1894.685	106.1496	0	20.75682
300001	1338.533	77.80388	0	15.74228
300002	1499.098	113.8098	0	2.933944
300003	2244.167	95.32138	0	22.1886
300004	1651.571	96.23018	0	21.38308
300005	1716.298	75.41242	0	12.06339
300006	2035.504	107.9095	0	14.01696
300007	1866.034	91.99005	0	5.311748
300008	2080.056	79.72078	0	12.56448
300009	1743.45	95.25403	0	10.19433
300010	1544.347	94.66212	0	30.65041

300011	758.8917	73.68779	0	11.66228
300012	1292.699	108.2895	0	22.82227
300013	1899.045	85.35335	0	9.359764
300014	1692.903	120.5142	0	23.22099
300015	2545.127	73.6724	0	14.2775
300016	1528.678	94.71427	0	8.307982
300017	1957.532	126.6271	0	16.10521
300018	1934.04	103.4774	0	9.900039
300019	1762.413	128.8249	0	4.049775
300020	1582.349	72.71487	0	10.45227
300021	1744.419	93.17659	0	17.04964
300022	1628.187	82.70443	0	4.077804

Table 4: Part Data

Appendix B

Proof that the iterations of the objective function provided a statistically reliable fitness value. Each iteration returns a health for a 12 quarter simulation.

Iteration #0	5612256
Iteration #1	5570142
Iteration #2	5612264
Iteration #3	5612315
Iteration #4	5612141
Iteration #5	5612147
Iteration #6	5612244
Iteration #7	5612170
Iteration #8	5570000
Iteration #9	5612170
Iteration #0	5562370
Iteration #1	5562333
Iteration #2	5562446
Iteration #3	5520251
Iteration #4	5562450
Iteration #5	5562282
Iteration #6	5520226
Iteration #7	5562540
Iteration #8	5520214
Iteration #9	5562375
Average	5576767
StDev	33384.15
fraction	0.005986

Table 5: Objective Function Statistics... Iteration and Health

Appendix C:

Sample of a solution, this is a partial listing of spare parts contained in a spares warehouse. The actual solution was much longer than recorded

300404
300407
300135
300974
200031
300966
200074
300852
300332
200031
300630
300999
200028
300138
300679
300819
300342
300535
300397
300734
300469
200040
300282
200067
300882
200031
300556
300506
300917
300029

Table 6: Sample of a Solution...