Evolutionary Algorithm for Airline Crew Scheduling

Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY, U.S.A.

June 10, 1999

Abstract

Crew Scheduling is one of the important problems in airline industry and has received much attention from the Operations Research community [1]. This problem is an NP-hard[2] constrained combinatorial optimization problem. We propose an Evolutionary Algorithm (EA) to solve this problem. A new graph based representation utilizes memory effectively, unlike previous attempts [3, 4, 5], and provides a framework in which we can easily develop various genetic operators. It is easy to insert the necessary constraints into the EA to make sure that the obtained schedule is valid.

1 Introduction

Crew scheduling is the assignment of the flight and training activities schedule for some period of time to each crew member [6]. A common terminology in the industry can be summarized as follows:

- A *flight leg* (*leg*) is a non-stop flight from A to B. *One-stop service* correponds to two flight legs, such as from A to C and C to B. *Connecting service* means that a change of aircraft is necessary.
- A city where crews reside is called a *base city* or *crew base*.
- A *duty period* (on-duty) is a sequence of flight legs. It can be thought of as describing a day's work for some crew.
- A *rotation* (*pairing*) is a sequence of *duty periods* interspersed by *layovers* (off-duty), which are long rest periods, intended for sleeping. Rotations start from and return to a crew base, and may last one or more days depending on companies' policies.
- A *deadheading* refers to the transport of crew members from one city to another as passengers.

Crew utilization has significant revenue return in the airline companies as crew costs constitute the largest direct operating cost of airlines next to fuel costs[7]. For example, in 1991, American Airlines reported spending \$1.3 billion on total crew cost (including salaries, benefits, and expenses) [7]. The corresponding figure for Nortwest Airlines (1989) has been reported to be \$1.05 billion and United Airlines (1993) spent \$0.6 billion just on pilots [8]. Hence, relatively small improvements in efficiency leverage into large cost savings [9].

Crew scheduling is a difficult combinatorial optimization problem [10]. This problem can be transformed to the Set Cover Problem or the Set Partition Problem. In this process, the flights are interpreted as tasks to be carried out and possible solutions for some tasks are produced. A binary matrix is obtained, whose rows represent the tasks and columns represent possible sub-solutions, with matrix entries indicating whether a task is assigned to a crew. Columns are generated by a complete enumeration or using heuristic methods. If n flight legs are to be covered, then the number of different producible columns is $O(n \cdot 2^{n-1})$. However, some columns are not feasible solutions due to FAA regulations, company policies, and labor union's requirements.

The schedule must contain a column set that covers all the flight legs and minimizes the total cost. The size of this column set will be less than n (the number of flights). Therefore, if we want to enumerate all the column sets and check every one of them, the number of cases we need to inspect is

$$\sum_{i=1}^{n} \binom{O(n \cdot 2^{n-1})}{i}$$

Crew Assignment (Bidline/Rostering) is the task of assigning individuals to the pairings. The objective is to cover all work as well as training requirements, vacations, etc., while satisfying work regulations and minimizing the crew cost. If crew members are paid a fixed salary (which is usually the case in Europe) one tries to minimize the number of crew members.

The crew cost depends on the quality of the solution to the pairing problem as well as the assignment problem, but since it is not possible to make up for poor pairings in the assignment process, it is reasonable to expect that savings in the pairings process will lead to savings in total crew costs [11].

1.1 Constraints

The Federal Aviation Administration (FAA) has established a complex set of limitations designed to ensure that crew members can fulfill their duties without significant risk of degradation of performance due to fatigue. Each pairing must conform to the limitations of the FAA and the published work rules in force at the airline.

1. Each flight must be covered by at least one crew.

2. Each pairing must begin at a crew base and return to the same base.

3. A duty period is usually restricted to eight hours of flying and 13 hours of total duty time, including briefing and debriefing. For night flying, a maximum duty time can not exceed ten hours.

4. The minimum rest period between duty periods is typically ten (10) hours. The minimum rest period is typically twelve hours when more than 8 hours flying are scheduled in a 24 hour period.

5. The flying time in a duty period should exceed some percentage of the total time on duty, say 50%. This discourages the practice of scheduling long sit periods between flights. Similarly, the total flying time in the entire trip should exceed some percentage of the total hours.

6. Layover time cannot exceed a prespecified maximum amount (typically 36 hours) to ensure that the crew returns his/her home base within the maximum rotation time.

7. Crew assignments must consider the number of crews available at the crew bases and try to distribute the workload uniformly (*crew balance*). The capacity of each base is usually expressed in term of the minimum and maximum number of flying hours per month.

1.2 Components Of Cost

1. At most major airlines, union contracts specify that flight crews will be guaranteed pay for some number of hours each day or each trip. The difference between the number of hours for which a crew member is paid and the number of hours he or she actually flies (*Pay and credit*) expresses the utilization for a particular crew. Long or frequent sits within a duty period, long overnight rests between duty periods, and *deadheading* are the main causes of excessive cost.

2. Crews are paid extra whenever

   (a) crews fly *onerous* night-flying;

   (b) rotations begin at one coterminal and end at another (coterminals are two or more airports servicing the same city);

   (c) consecutive flight segments in a rotation have less than ninety minute break between them;

   (d) they fly more than five hours consecutively (they are qpaid at a predetermined overtime rate);

   (e) their total time away from the base exceeds a specified amount.

3. Hotel and per-diem expenses increase if layovers are scheduled away from the home base.

Because of these complex rules, airline companies attempt to

1. keep crews on the same plane during a duty period as much as possible,

2. eliminate any pairing with cost (or cost per flight leg) greater than a certain threshold, and

3. prefer shorter pairings (no more than three days) since longer pairings would cause greater difficulty in rescheduling if the weather or other factors cause a pairing to be disrupted.

2 Previous Studies

Airline crew scheduling can be reduced to the Set Covering Problem (SCP) if dead-heading is allowed, or the Set Partition Problem (SPP) otherwise. Both are well-known combinatorial optimization problems. But relaxing the constraints and applying LP to the problem does not give satisfactory solutions because the size of the problem is a primary factor determining how good a solution can be obtained by LP for an integer problem. The size of the problem is huge and it is not practical to apply LP to the problem directly. Therefore cutting-plane, branch-and-cut[12] and branch-and-bound algorithms have been proposed to find an optimum solution.

The crew assignment problem fits nicely into a classical integer linear programming (ILP) formulation [13]. However, solving the ILP is relatively the easy part; like most real-world problems, the biggest problem is in accommodating all of the pay guarantees and work rule limitations into the matrix before the ILP is solved.

A branch-and-cut technique for this problem has been proposed and applied by Hoffman and Padberg [12]. They were able to solve the AA4 (a hard set partitioning problem, with 7195 columns and 426 rows) in 38.7 hours, proving the optimality of the solution.

Levine [3] tried to solve the problem using a GA. Levine also applied a local heuristic with the GA to solve the SPP. A solution to the SPP typically contains mostly zeroes. Therefore, Levine initialized the population according to this observation. The GA with hybrid strategy did not give optimum results. The branch-and-cut based algorithm performed better than the GA.

Chu [5] preferred to use fitness and *unfitness* of the solution instead of one fitness value to handle the infeasible solutions obtained by the GA. He used the logical OR crossover. Let $P_{1,i}$ and $P_{2,i}$ represent the i^{th} genes of the selected parents. Then the i^{th} gene of the offspring (C_i) is $P_{1,i} \vee P_{2,i}$. He used an adaptive mutation rate and matching selection of the parents to apply crossover. In this selection approach, the first parent is selected by tournament selection and the second parent is selected according to a maximum compatibility score. This scores all the solutions according to the first parent and selects the one with maximum score. Chu attempts to select the two parents such that they cover as many rows as possible. He used the ranking replacement strategy and divided the fitness-unfitness surface into four regions according to the child's (fitness-unfitness) value: high fitness-high unfitness, low fitness-high unfitness, high fitness-low unfitness, and low fitness-low unfitness. He considers the high fitness-high unfitness region first and if he finds something in that region he replaces this solution with the new child. Otherwise, the remaining regions are successively examined. A reduction procedure is applied before applying the GA. According to his results, the GA is outperformed by the conventional CPLEX Mixed Integer Solver package and the bound-and-cut algorithm [4].

3 Graph Representation

Graph representation is a convenient way for modeling many different problems such as minimum flow, shortest path, and satisfiability problems. In our work, a new directed graph representation is used to model the compatibility relationship between flights, and algorithms are applied using this representation.

It is preferable to use the flight schedule instead of pre-processed columns as input data. After all, column generation techniques try to keep the best columns but some suboptimal (not the best) elements are needed to get a better solution in terms of global optimization [14, 15]. Therefore, our proposed solution technique starts from the flight schedule and builds the flight graph to take care of some problem specific constraints.

In our representation, the i^{th} node denotes the i^{th} flight. The directed edge (i,j) denotes that $flight_j$ can be flown after $flight_i$. This edge carries information requiring the time and city constraints. Other constraints, which cannot be expressed in this graph, are handled in the algorithm. Theoretically, there can be at most $(\frac{n^2-n}{2})$ edges on this graph for n flights. $(n-1)$ edges will be chosen to produce the necessary schedule, attempting to minimize the total cost as much as possible. Table 1 and Figure 1 illustrates the graph representation, using the contsraint that there is at least 30 minutes break between consecutive flights.

As a simplification for the problem, it is assumed that any crew members may be assigned to any flights, i.e., this is a single fleet problem. This assumption does not excessively oversimplify the actual problem since the same method can be applied

Flight Number	Dep. City	Des. City	Dep. Time	Arr. Time
0	1	5	07:00	09:30
1	5	1	09:30	11:30
2	1	6	10:15	11:30
3	1	2	12:00	13:30
4	6	3	12:30	14:30
5	2	1	14:10	15:20
6	3	6	15:30	21:45
7	1	0	16:00	16:45
8	0	1	17:30	18:15
9	1	5	19:00	20:30
10	5	1	21:30	23:00
11	6	1	22:30	23:45

Table 1: A sample schedule.

for each crew type and for each fleet type, sequentially.

The goal can be summarized as the following:

- respect constraints (each rotation starts and ends at the same base city, the time between flights is greater than the allowed threshold, etc.),
- minimize number of crews,
- maximize crew time utilization, and
- balance workloads between crews.

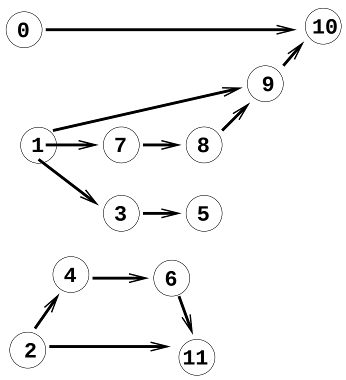

Figure 1: Directed graph representation for the flight schedule in Table 1.

4 Algorithms Based on Graph Representation

It is possible to define various algorithms based on the graph representation to obtain several solutions. First we define two greedy algorithms that require backtracking and do not guarantee that the result will be feasible.

4.1 Greedy algorithm based on picking edges in sequence

When the flights are sorted according to departure time, we observed that edges connecting flights appear in the upper triangular portion of the adjacency matrix. The cost matrix is defined based on the time between flights. Starting from the last column, the algorithm walks back to the first column. For each uncovered flight (f), it tries to find a path to add this flight at the beginning of the path. If it finds such a path, then it adds flight f. If it does not find such a path and the destination of flight f is a base city, then it finds the minimum cost cell on this column. The row index (r) of this cell denotes that flight r can precede flight f. The algorithm defines each path based on this information. If there is no such cell on the column and the destination of flight f is a base city, then it defines a new path containing flight f.

Since this algorithm does not guarantee that all rotations start and end at the same base city, heuristic backtracking (see Section 5.4) is applied to correct the schedule.

4.2 Greedy Algorithm Based on Bin Packing

The problem can be expressed as a minimum bin packing problem. The capacity of each bin is known. Assignments must be packed into bins while respecting the schedule constraints and minimizing the number of required bins.

Of course, the minimum bin packing problem has less constraints than this version, therefore it requires extra caution to enforce the constraints while packing items into bins.

The algorithm assumes that flights are sorted in decreasing order with respect to departure times. It starts from the first element and walks through the array. For each flight, the algorithm looks at the currently defined bins and checks whether the flight can be pushed into any one of available bins. If there is more than one bin which can accept the flight, then the one with minimum workload is preferred. If there is no bin that can accept this flight, then a new bin is defined and the current flight is pushed into this bin.

The algorithm defines a rotation from each bin and applies feasibility checks. In case the rotation is not feasible, it attempts to find the longest feasible subpath. Then it applies HeuristicPatchUp (see Section 5.4) and AddEdgesForPatchUp (see Section 5.6) algorithms.

5 Genetic Algorithm for ACS

For the GA, this section describes the chromosome representation, the distance measure between two chromosomes, the fitness value of a chromosome, initialization of the first population, and then genetic operators.

5.1 Chromosome Representation

Each chromosome contains rotations (pairings) for each crew. Each crew assignment contains a set of rotations. Each rotation consists of duty-periods that represents a path in the flight graph. The minimum number of crews is not specified so that the algorithm tries to minimize the number of crews. We may define a lower limit to ensure that the algorithm does not reduce the number of crews below this threshold while distributing the workload.

The Hamming distance between two chromosomes (c_i and c_j) is defined in terms of the number of different edges. Let $E(c_i)$ denote the set of edges used by chromosome c_i, then the *distance* between chromosome c_i and c_j can be described as follows:

$$d(c_i, c_j) = \frac{|E(c_i)| - |E(c_i) \cap E(c_j)|}{|E(c_i)|}$$

Note that for all feasible solutions, $|E(c_i)| < n$ and $|E(c_j)| < n$.

5.2 Fitness Function

For each chromosome, the following constraints are enforced to ensure feasibility:

- There must be a minimum time delay between consecutive flight legs in a rotation.
- Each rotation must start and end at the same base city.
- If the length of the duty period is less than 8 hours, then the next duty period must start 8 hours later.
- If the length of the duty period is greater than 8 hours and less than 13 hours, then the next duty period must start 12 hours (720 minutes) later.

The cost includes overnight stay, deadheadings, and overtime pays. Each crew gets paid a minimum amount of money to fly a certain amount of time for a given period. It is the responsibility of the company to utilize this time. If the schedule assigns unbalanced rotations, then the company gets punished from two sides: overtime and underutilized personnel time. If a crew flies more than what she/he gets paid for, then extra bonus must be paid for the overtime. Therefore, the goal is to build a schedule which has balanced rotations, utilizes the personnel's paid time and minimizes the overtime and overnight payments.

5.2.1 Cost of Solution

A cost value of rotation is calculated by adding the amount paid to crew and hotels for overnight stays. Using the variables described in Figure 2, the following criteria are applied when evaluating each rotation.

- No duty can be longer than $MaxDutyTime$ and cannot contain more than $MaxFlyingTime$ minutes of flying.
- If a duty contains less than $MaxFlyingTime$ minutes of flying then a crew get paid at $PayForMinPaidFlyingTime(PFMPFT)$.
- If a duty contains more than $MinPaidFlyingTime(MPFT)$ minutes of flying (tft) then a crew earns

  $$PFMPFT + (tft - MPFT) \times EBFOT$$

- If a crew stays overnight in another place different from his/her domicile, then for each overnight stay

  $$HE + PDE \times LengthOfStay$$

  will be charged to this rotation.
- If the length of a duty (ld) is longer than $AveDutyTime$, then the minimum required rest time ($MRRT$) is calculated as

  $$\frac{(ld - ADT)}{(MxR - MnR)} \times (MxDT - ADT) + MnR$$

$MaxRotationTime(MxRT)$	=	Maximum rotation time in minutes. (3*24*60 minutes)
$MaxDutyTime(MxDT)$	=	Maximum duty time in minutes. (14*60 minutes)
$AveDutyTime(ADT)$	=	Average duty time in minutes. (8*60 minutes)
$MaxFlyingTime(MxFT)$	=	Maximum flying time in minutes. (11*60 minutes)
$MinPaidFlyingTime(MPFT)$	=	Minimum paid flying time in minutes. (5*60 minutes)
$PayForMinPaidFlyingTime(PFMPFT)$	=	Minimum payment for a duty. (300)
$ExtraBonusForOverTime(EBFOT)$	=	Bonus for overtime flying for each extra minute. (1.5)
$HotelExpense(HE)$	=	Hotel expense. (30)
$PerDiemExpense(PDE)$	=	Additional pay for each minute when crew stays away from his/her domicile. (0.1)
$MinRest(MnR)$	=	Minimum rest time between duties in minutes. (8*60 minutes)
$MaxRest(MxR)$	=	Maximum rest time when the previous duty is as long as $MaxDutyTime$ in minutes. (13*60 minutes)
$CostPerMinuteUtilization(CPMU)$	=	the loss to the company when one minute of a crew time is wasted. (0.2)

Figure 2: Variables used in the problem to calculate the cost of each schedule, with values used in our simulations.

where MxR, MnR, and ADT conforms with the standard values used in the industry.

- If the length of rest (RT) between two duties is longer than minimum required rest time ($MRRT$), then this underutilized crew time is added as an underutilization cost (UUC):

$$UUC = (RT - MRRT) \times CPMU$$

- In summary the components of cost for a sequence of rotations assigned to a crew are:
  - underutilized time between rotations,
  - underutilized time between duties in each rotation,
  - hotel and per-diem expenses for each rotation, and
  - required pay for each duty in each rotation.

Each rotation starts from and ends at the same base city and there is a minimum time requirement between consecutive flights (30 minutes). After the cost of schedule is calculated, a fitness value is assigned to this solution. The fitness value is defined as:

$$\frac{1}{scost(1+\frac{n_{ef}}{n_f})(1+\frac{n_{sf}}{n_f})(1+\frac{n_{uf}}{n_f})}$$

where

$$scost = 1 + \frac{cost - minCostInPop}{maxCostInPop - minCostInPop}$$

,

$minCostInPop$	=	the minimum cost value in the current population,
$maxCostInPop$	=	the maximum cost value in the current population,
n_f	=	the number of flights in the problem,
n_{ef}	=	the number of rotations that does not end where it is started,
n_{sf}	=	the number of rotations that does not start from a base city, and
n_{uf}	=	the number of uncovered flights.

5.3 Initialization

The initialization phase applies the following steps repeatedly to obtain the initial population.

- Let E' be the current graph.
- Pick all edges coming to nodes with outdegree 0, from E' and place them into set E_s.
- As long as there are edges in E_s and there are un-
covered nodes, apply AddEdgesFromEdgeSet (see Section 5.5) and update E' by removing edges incident on the covered nodes. Update E_s according to the new graph E'.
- If there are uncovered edges, then apply HeuristicPatchUp (see Section 5.4) algorithm, which applies backtracking to fix the current solution.

- If there are uncovered edges, then apply AddEdgesForPatchUp (see Section 5.6) algorithm, which attempts to cover uncovered nodes by relaxing the constraints.

5.4 HeuristicPatchUp

Some choices in the algorithm make some nodes inaccessible. To be able to cover those nodes, this algorithm puts back some edges into the current graph and applies **AddEdgesFromEdgeSet** (see Section 5.5) algorithm again. This performs *backtracking* on the schedule. The number of times the algorithm is allowed to backtrack is given as a parameter. The algorithm can be improved with the abandoned list technique of Tabu Search to prevent putting back edges removed recently.

5.5 AddEdgesFromEdgeSet

This algorithm tries to build paths by concatenating edges backwards. The edge set is given as a parameter with the current graph since the graph changes continuously. It builds paths which are not longer than a given number of hours (a parameter) and checks their feasibility. If a path is not valid, then it looks at the maximum feasible subpath and adds this to the solution. It removes all the edges related to new covered nodes from the graph and the current edge set. Then it continues to search another path until it cannot find any other path by using the current edge set.

5.6 AddEdgesForPatchUp

This algorithm tries to cover uncovered nodes by relaxing the constraints. The first relaxation is to allow deadheading. In this phase, it tries to cover uncovered nodes by using already covered nodes. In the second phase, if there are still uncovered nodes, then it tries to cover them with paths without checking their feasibility.

5.7 Genetic Operators

The mutation operator modifies duty periods by cutting and splicing them. The crossover operator works at the rotation level and collects duty periods into various rotations.

5.7.1 Mutation Operator

This operator selects two edge lists (l_i and l_j), cuts both of them at some suitable point, and splices together the pieces.

For example, let $l_i = v_0, v_1, ..., v_k$ and $l_j = v'_0, v'_1, v'_2, ..., v'_k$. Assume that there are edges from v_i to v'_j and v'_{j-1} to v_{i+1}. Then the resulting edge lists will be $l_i = v_0, v_1, ., v_i, v'_j, ..v'_k$ and $l_j = v'_0, v'_1, v'_2, .., v'_{j-1}, v_{i+1}, ...v_k$.

5.7.2 Crossover Operator

We have experimented with three crossover operators. The ıSet based operator tries to use genes inherited from both parents as much as possible. The ıTime based operator cuts the schedule into two pieces and recombines them. The ıDistance preserving operator tries to preserve genes common to both parents but fills up the rest using alleles not used in any parent.

Set Based Crossover (SPX)

- Take the common edge lists and place them into the offspring.
- Put the edge lists coming from the parents into two different sets, A_1 and A_2.
- Produce a set (C) consisting of the partial matching between elements in A_1 and elements in A_2, and rank them according to the number of edges they contain.
- Pick an element from the set C and place it in the offspring. Go through the rest of the set C and for any element that has the same edge as the recently picked element, either split this edge list (if it has more than one edge) or cancel it (if there is only one edge) from the set C.

Time Based Crossover (TPX)

- Flights are stored in the increasing order of take off time. Randomly choose a flight (f).
- For two parents A and B, divide rotations in each schedule according to the arrival time of the randomly chosen flight f; let the splitting separate A into (A_1,A_2), and B into (B_1,B_2).
- We need to combine A_1 with B_2 and B_1 with A_2 for the crossover operation. The combination phase applies the following:

- For each rotation (r_1) in A_1, attempt to find a rotation (r_2) coming from B_2 such that the destination city of r_1 is same as the starting city of r_2.
- For each rotation r in B_2,
  * Try to merge r with the current solution.
  * Try to add each flight from r to the current solution. If a flight can merge with any rotation in the current solution, then it will inserted into that rotation. Otherwise, it will be introduced as a new rotation.

- B_1 and A_2 are combined in similar manner.

Distance Preserving Crossover Operator (DPX)

Distance preserving operators try to produce an offspring at equal distance from both parents. Since the solution consists of edges, a distance measure based on edges can be defined and the following algorithm can be applied to preserve this distance:

- Take the common edge lists and place them into the offspring.
- Build an abandoned edge matrix (A_{ij}) where $A_{ij} = 1$ if an arc from v_i to v_j is in the edge list of a parent but not in the offspring.
- Try to patch up the offspring without using abandoned edges.
- If it cannot be repaired, then allow the repair algorithm to use all edges in graph to repair the offspring.

5.8 Local Search

Local Search is applied after recombination and mutation operators. Local search consists of various steps:

- All the rotations are extracted and a greedy bin packing approach is applied. In this approach, first all the rotations are sorted based on the take off time. Then, rotations are packed into bins. If there is a time overlap between the current rotation and the bins, then a new bin is introduced with this rotation. If there is no time overlap and there are several bins that can accept the current rotation then the bin with minimum time gap is preferred.
- After the re-organization process is complete, then the 1PX and 2PX crossover operations are applied on the rotations of the solution until no further improvement is observed. Offspring of 1PX/2PX replace parent if they have higher fitness value.

5.9 Restart

The following restart procedure is applied periodically after $\frac{M}{10}$ iterations, where M is the maximum number of iterations.

1. Sort the chromosomes based on their fitness values.
2. Start from the best individual and compare the best with the other individuals by using a hamming distance (HD) measure. If HD is less than 10%, then the chromosome is marked for random restart.

   Repeat this step with remaining unmarked chromosomes in decreasing fitness order.
3. Apply random initialization on each marked chromosome.

5.10 Restoration

A restoration algorithm is required when the solution does not cover all the flights. The algorithm we used randomly chooses from previously found rotations to restore the solution or uses HeuristicPatchUp (see Section 5.4) algorithm. To be able to keep the best found subsolutions, the system defines a rotation bag for each flight where the elements are updated by using ıcost/total_flying_time value. The set of rotations that give the better value replace the worst solution in the bag. This process is applied on each solution obtained without using pre-calculated subsolutions.

6 Results - needs rewrite

Test data sets were obtained by querying various airline Web sites and filtering the flight data for one fleet. To be able to compare the results with the Set Cover Problem solution techniques, we enumerated [16] all columns whose total time duration is ≤ 3 days. The specifics of each problem is given in Table 34 where the first column gives the number of flights, the second gives the number of columns produced with enumeration and the third column gives the time span of each schedule. Greedy algorithms are developed based on choice of Chavatal, Bäck and our new heuristic which multiplies Bäck's cost-ratio with the number of overlaps between the current cover and the cover of the columns. Similarly, a GA was implemented for the Set Cover Problem.

The GA based on a directed graph representation was better than other approaches in many respects:

- It distributes workload better among crew members since the standard deviation of rotation times for each problem is the smallest for all four problems.
- It offers results with smaller number of crews, e.g., the 102 flight problem (see Table 3).
- It results in fewer deadheadings, hence the assigned flying time for crews, yielding better cost results.

The only problem with the GA is the computation time requirement. The GA did not yet converged, suggesting that that with longer runs the algorithm can produce better solutions as we observed for the 102 flight problem. New operators may result in better time requirements.

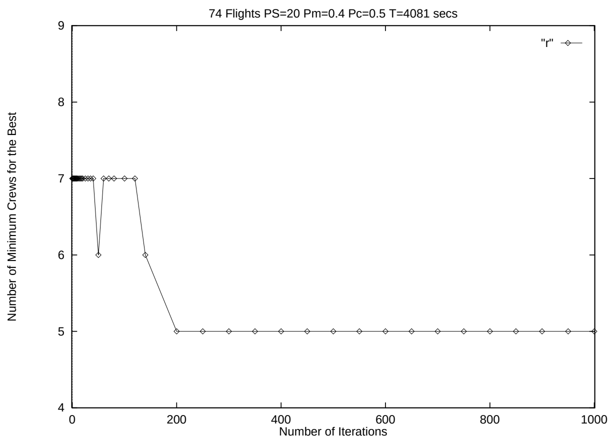

Figure 3: 74 Flights

7 Future Work

Experiments with larger problems are under way. The directed graph representation can be applied

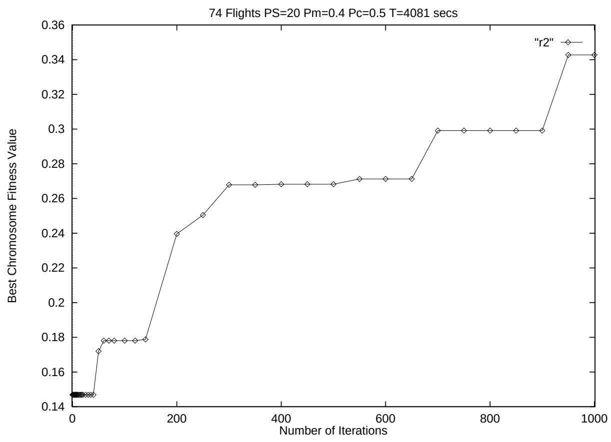

Figure 4: 74 Flights

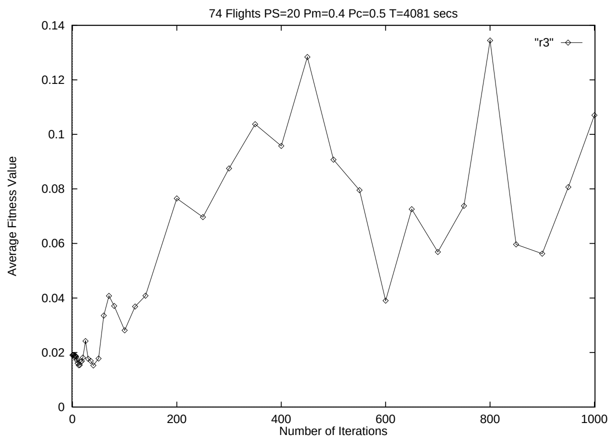

Figure 5: 74 Flights

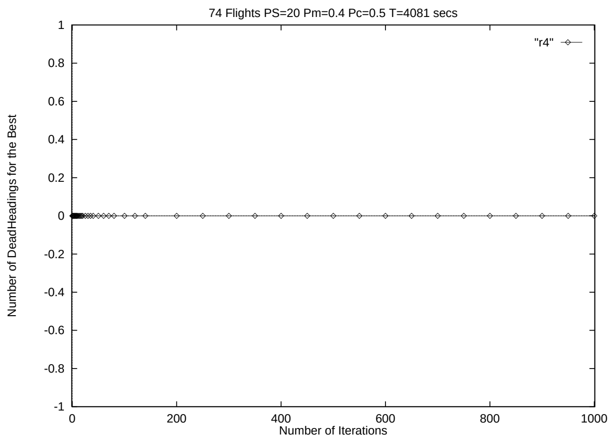

Figure 6: 74 Flights

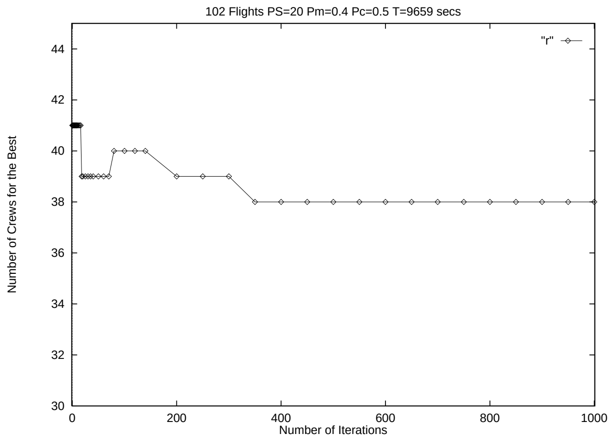

Figure 7: 102 Flights

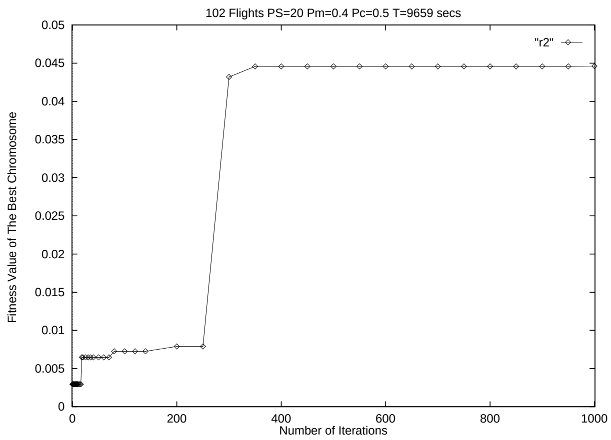

Figure 8: 102 Flights

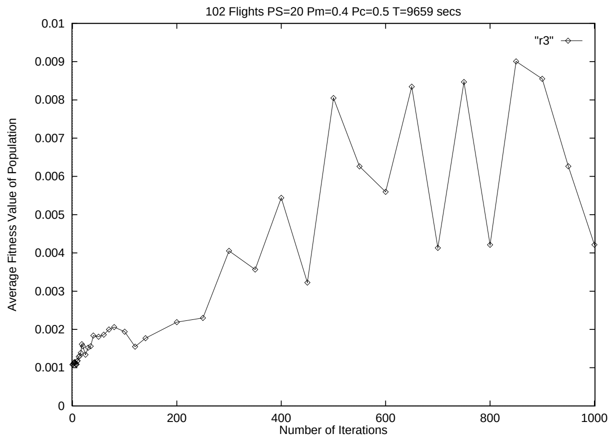

Figure 9: 102 Flights

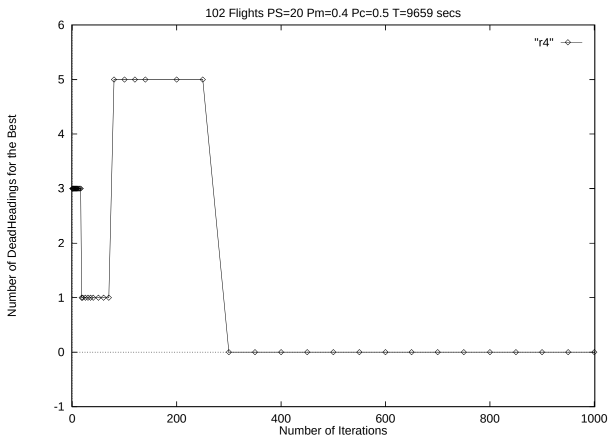

Figure 10: 102 Flights

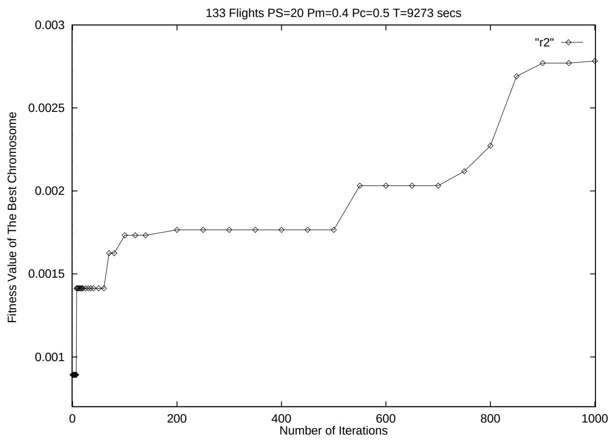

Figure 12: 133 Flights

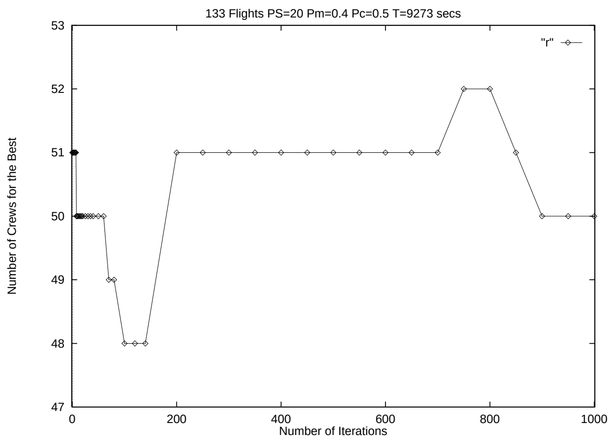

Figure 11: 133 Flights

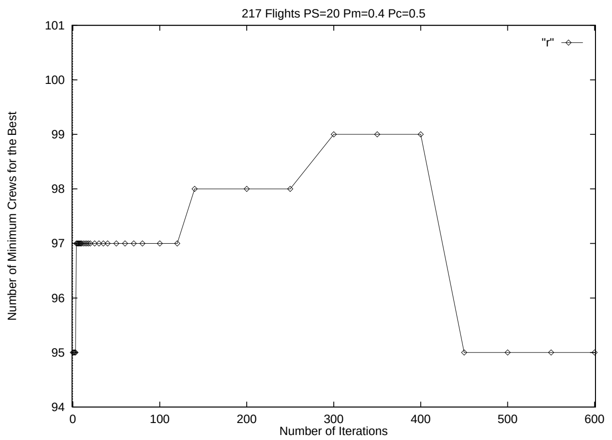

Figure 13: 217 Flights

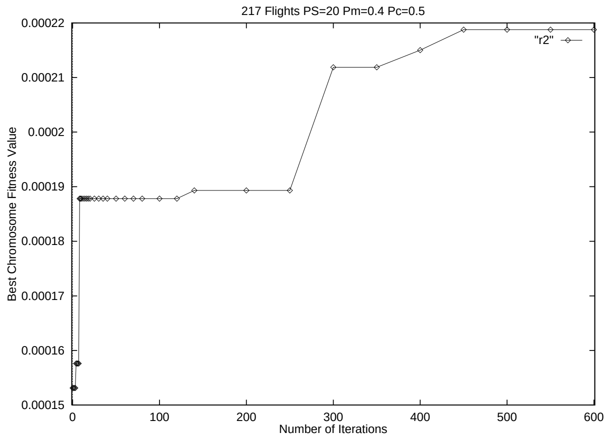

Figure 14: 217 Flights

to several other problems such as vehicle scheduling and bus driver scheduling problems.

We are working on a GA based column generation approach for similar problems where we defined a multi objective genetic algorithm to obtain a better and compatible column set.

References

[1] J.P. Arabeyre, J. Feanley, F.C. Stieger, and W. Teather. The Airline Crew Scheduling Problem. *Transportation Science*, (3):140–163, 1969.

[2] M.R. Garey and D.D. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. Freeman, San Fransisco, California, 1979.

[3] D. Levine. *Application of a Hybrid Genetic Algorithm to Airline Crew Scheduling*. PhD dissertation, IIT , Chicago, USA, Computer Science Department, 1995.

[4] P.C. Chu and J.E. Beasley. A Genetic Algorithm for the Set Partitioning Problem. Technical report, Imperial College, UK, 1995.

[5] P.C.H. Chu. *A Genetic Algorithm Approach for Combinatorial Optimisation Problems*. PhD dissertation, Imperial College, , UK, Computer Science Department, 1997.

[6] F. Giannessi and B. Nicoletti. The Crew Scheduling Problem: A Travelling Salesman Approach. *in [10]*, 1979.

[7] R. Anbil, E. Gelman, B. Patty, and R. Tanga. Recent Advances in Crew-Pairing Optimization at American Airlines. *INTERFACES*, 21(1):62–74, 1991.

[8] E. Addersson, E. Housos, N. Kohl, and D. Wedelin. Crew Pairing Optimization. *in [11]*, pages 228–257, 1998.

[9] R. Anbil, R. Tanga, and E. L. Johnson. A global approach to crew-pairing optimization. *IBM Systems Journal*, 31(1):71–78, 1992.

[10] N. Christofides, A. Mingozzi, P. Toth, and C. Sandi. *Combinatorial Optimization*. John Wiley & Sons, Ltd., 1979.

[11] G. Yu (Editor). *Operations Research in Airline Industry*. Kluver Academic Publishers, Boston/Dordrecht/London ISBN 0-7923-8039-8, 1998.

[12] K. L. Hoffman and M. Padberg. Solving Airline Crew Scheduling Problems by Branch-and-Cut. *Management Science*, 39(6):657–682, 1993.

[13] I. Gershkoff. Optimizing Flight Crew Scheduling. *INTERFACES*, 19(4):29–43, 1989.

[14] L. J. Fogel. Top-DOwn Evolutionary Engineering. In X. Yao, J-H. Kim, and T. Furuhashi, editors, *First Asia-Pacific Conference on Simulated Evolution and Learning(SEAL'96)*, ISBN : 3-540-63399-5, 1997. Springer-Verlag.

[15] J-M. Rousseau and J. Desrosiers. Results obtained with Crew-Opt: A Column Generation Method for Transit Crew Scheduling", Driver Scheduling. In J.R. Daduna, I. Branco, and J.M. Paxiao, editors, *6th International Workshop on Computer-Aided Scheduling of Public Transport(1993)*, pages 349–358, ISBN : 3-540-60193-7, 1995. Springer-Verlag.

[16] E.M. Reingold, J. Nievergelt, and N. Deo. *Combinatorial Algorithms Theory and Practice*. Prentice Hall, 1977.

[17] V. Chavatal. A Greedy Heuristic for the Set Covering Problem. *Mathematics of Operations Research*, 4(3):233–235, 1979.

[18] T. Back, M. Schutz, and S. Khuri. A Comparative Study of a Penalty Functions, a Repair Heuristic, and Stochastic Operators with the Set-Covering Problem. In J.-M. Alliot, E. Lutton, E. Ronald, M. Schoenauer, and D. Snyers, editors, *Artificial Evolution, European Conference, AE 95*, pages 320–332. Springer-Verlag, 1995.

[19] P.C. Chu and J.E. Beasley. Constraint Handling in Genetic Algorithms: the Set Partitioning Problem. Technical report, Imperial College, UK, 1997.

	Chavatal [17] SCP	Back [18] SCP	SCP based [19] CBGA	Greedy#3 SCP	The Best with GA
Number of Crews	8	7	7	10	8 - 7
Average Rotation Time	1067.75	1229.57	1126	895.2	1062.12 - 1126
STD of Rotation Times	431.183	539.94	663.011	536.112	414.628 - 527.059
Number of Dead-headings	6	0	0	10	0 - 0
Not ends where it started	0	0	0	0	0 - 0
Total Flying Time (mins)	8542	8607	7882	8952	8497 - 7882
Cost	22636.8	15431.9	17085	19420.9	12321.5 - 12510.8
Running Time (sec)	208	58	24787	150	8515

Table 2: Results for 74 flights problem. 3 day is used as a maximum rotation time. The sum of flying times is 7882 minutes. CBGA is run for 1000 generations. Graph based Steady State GA run with 20 population size, 0.5 crossover probability, and 0.4 mutation probability for 400 iterations. The result shows two example solution from the final population.

	Chavatal [17] SCP	Back [18] SCP	SCP based [19] CBGA	Greedy#3 SCP	The Best with GA
Number of Crews	55	42	41	42	39
Average Rotation Time	242.8	250.667	243.488	216.762	206.625
STD of Rotation Times	57.673	62.507	56.010	79.545	73.285
Number of Dead-headings	71	34	27	13	0
Not ends where it started	0	0	0	0	0
Total Flying Time (mins)	13354	10528	9983	9104	8058
Cost	16840	13272	12592	13310	12210
Running Time (sec)	< 1	3	412	5	5716

Table 3: Results for 102 Flights problem. 3 day is used as a maximum rotation time. The sum of flying times is 8058 minutes. CBGA is run for 1000 generations. Graph based Steady State GA run with 20 population size, 0.5 crossover probability, and 0.4 mutation probability for 1000 iterations.

	Chavatal [17] SCP	Back [18] SCP	SCP based [19] CBGA	Greedy#3 SCP	The Best with GA
Number of Crews	64	48	49	47	48
Average Rotation Time	292.703	331.75	316.673	326.76	299.542
STD of Rotation Times	88.462	95.233	98.042	105.56	109.35
Number of Dead-headings	58	24	19	15	4
Not ends where it started	0	0	0	0	0
Total Flying Time (mins)	18733	15924	15517	15358	14378
Cost	23806.9	20335.9	20043.1	20365.8	19295.3
Running Time (sec)	< 1	8	1527	17	9470

Table 4: Results for 133 flights problem. Maximum rotation length is set to 3 days. The sum of flying times is 13958 minutes. Graph based Steady State GA run with 20 population size, 0.5 crossover probability, and 0.4 mutation probability for 2000 iterations.

	Chavatal [17] SCP	Back [18] SCP	SCP based [19] CBGA	Greedy#3 SCP	The Best with GA
Number of Crews	132	88	96	89	91
Average Rotation Time	245.742	271.614	275.208	244.978	232.099
STD of Rotation Times	48.144	67.343	78.114	74.930	77.3252
Number of Dead-headings	201	82	115	51	45
Not ends where it started	0	0	0	0	0
Total Flying Time (mins)	32438	23902	26420	21803	21121
Cost	40398.6	29178.6	33340.6	29224.6	30006.6
Running Time (sec)	5	50	4348	164	124216

Table 5: Results for 217 flights problem. CBGA is run for 1000 generations. The sum of total flying times is 17614 minutes. Graph based Steady State GA run with 20 population size, 0.5 crossover probability, and 0.4 mutation probability for 800 iterations.

	The Best SXP	The Best TXP	The Best DPX
Number of Crews	7	7	7
Average Rotation Time	1126	1126	1126
STD of Rotation Times	477.374	440.007	369.516
Number of Dead-headings	0	0	0
Not ends where it started	0	0	0
Total Flying Time (mins)	7882	7882	7882
Cost	14995.8	15068	15249.8
Running Time (sec)	5844	5152	12119

Table 6: Comparison of varius crossover operations for 74 Flights problem. GA is run 1000 generations, population size is 20, mutation probability is 0.4, crossover probability 0.5.

	The Best SXP	The Best TXP	The Best DPX
Number of Crews	44	42	42
Average Rotation Time	183.136	196.048	191.857
STD of Rotation Times	54.557	66.498	69.057
Number of Dead-headings	0	2	0
Not ends where it started	0	0	0
Total Flying Time (mins)	8058	8234	8058
Cost	13012	12932	12922
Running Time (sec)	2942	3143	2439

Table 7: Comparison of varius crossover operations for 102 Flights problem. GA is run 400 generations, population size is 20, mutation probability is 0.4, crossover probability 0.5.

	The Best SXP	The Best TXP	The Best DPX
Number of Crews	61	59	62
Average Rotation Time	242.656	254.492	233.839
STD of Rotation Times	118.056	117.667	115.513
Number of Dead-headings	7	8	4
Not ends where it started	0	0	1
Total Flying Time (mins)	14802	15015	14498
Cost	22175.8	22324.5	23122.5
Running Time (sec)	17927	18259	18090

Table 8: Comparison of varius crossover operations for 133 Flights problem. GA is run 1000 generations, population size is 20, mutation probability is 0.4, crossover probability 0.5.

	Chavatal [17] SCP	Back [18] SCP	SCP based [19] CBGA	Greedy#3 SCP	The Best with GA
Number of Crews	6	6	7	6	7
Average Rotation Time	1326.33	1661.5	1424.14	1661.5	1424.14
STD of Rotation Times	495.03	420.14	480.769	536.112	804.665
Number of Dead-headings	9	0	0	0	0
Not ends where it started	0	0	0	0	0
Total Flying Time (mins)	11937	9969	9969	9969	9969
Cost	22609.7	17609	18546.3	17609	15992.1
Running Time (sec)	< 1	< 1	218	< 1	1561

Table 9: Results for MandarinAir problem (28 flights). The sum of flying times is 9969 minutes. CBGA is run for 1000 generations. Graph based Steady State GA is run 1000 iterations, population size is 20, mutation probability is 0.4, crossover probability 0.5.

	Chavatal [17] SCP	Back [18] SCP	SCP based [19] CBGA	Greedy#3 SCP	The Best with GA
Number of Crews	25	22	20	22	21
Average Rotation Time	294.4	316.591	348.25	316.591	325.952
STD of Rotation Times	100.701	101.044	96.853	101.044	117.945
Number of Dead-headings	12	4	4	4	4
Not ends where it started	0	0	0	0	0
Total Flying Time (mins)	7360	6965	6965	6965	6845
Cost	9636.67	8914.17	8790.33	8914.17	8975.33
Running Time (sec)	< 1	< 1	280	< 1	1620

Table 10: Results for OrlandoAir problem (54 flights). The sum of flying times is 6485 minutes. CBGA is run for 1000 generations. Graph based Steady State GA is run 1000 iterations, population size is 20, mutation probability is 0.4, crossover probability 0.5.

	Chavatal [17] SCP	Back [18] SCP	SCP based [19] CBGA	Greedy#3 SCP	The Best with GA
Number of Crews	19	20	18	20	18
Average Rotation Time	387.579	375.95	390.667	375.95	378.333
STD of Rotation Times	122.144	114.796	127.196	114.796	124.104
Number of Dead-headings	10	10	6	10	6
Not ends where it started	0	0	0	0	0
Total Flying Time (mins)	7364	7519	7032	7519	6810
Cost	9698	9730	9356	9730	8912
Running Time (sec)	< 1	< 1	318	< 1	4420

Table 11: Results for B727_May1_43 problem (43 flights). The sum of flying times is 5829 minutes. CBGA is run for 1000 generations. Graph based Steady State GA is run 1000 iterations, population size is 20, mutation probability is 0.4, crossover probability 0.5.

	Chavatal [17] SCP	Back [18] SCP	SCP based [19] CBGA	Greedy#3 SCP	The Best with GA
Number of Crews	26	24	23	24	23
Average Rotation Time	243.462	242.917	237.609	231.875	232.391
STD of Rotation Times	76.381	64.677	61.729	71.702	63.739 - 77.654
Number of Dead-headings	30	23	17	19	5
Not ends where it started	0	0	0	0	0
Total Flying Time (mins)	6330	5830	5465	5565	5345
Cost	8340	7540	7120	7553	7112
Running Time (sec)	< 1	< 1	265	1	4885

Table 12: Results for GBXSEA_M_55 problem (55 flights). The sum of flying times is 4415 minutes. CBGA is run for 1000 generations. Graph based Steady State GA is run 5000 iterations, population size is 20, mutation probability is 0.4, crossover probability 0.5.

	Chavatal [17] SCP	Back [18] SCP	SCP based [19] CBGA	Greedy#3 SCP	The Best with GA
Number of Crews	12	12	12	12	12
Average Rotation Time	152.5	152.5	152.5	152.5	152.5
STD of Rotation Times	50.682	50.682	50.682	50.682	50.682
Number of Dead-headings	0	0	0	0	0
Not ends where it started	0	0	0	0	0
Total Flying Time (mins)	1830	1830	1830	1830	1830
Cost	3600	3600	3600	3600	3600
Running Time (sec)	< 1	< 1	80	< 1	121

Table 13: Results for GBXORD_M_26 problem (26 flights). The sum of flying times is 1830 minutes. CBGA is run for 1000 generations. Graph based Steady State GA is run 400 iterations, population size is 20, mutation probability is 0.4, crossover probability 0.5.

	Chavatal [17] SCP	Back [18] SCP	SCP based [19] CBGA	Greedy#3 SCP	The Best with GA
Number of Crews	23	15	15	15	15
Average Rotation Time	146.522	160.667	160.667	152.667	152.667
STD of Rotation Times	78.052	81.442	78.779	84.968	71.317
Number of Dead-headings	20	4	4	2	2
Not ends where it started	0	0	0	0	0
Total Flying Time (mins)	3370	2410	2410	2290	2290
Cost	6900	4500	4500	4527	4500
Running Time (sec)	< 1	< 1	186	1	1326

Table 14: Results for GBXSFO_M_46 problem (46 flights). The sum of flying times is 2190 minutes. CBGA is run for 1000 generations. Graph based Steady State GA is run 1000 iterations, population size is 20, mutation probability is 0.4, crossover probability 0.5.

	Chavatal [17] SCP	Back [18] SCP	SCP based [19] CBGA	Greedy#3 SCP	The Best with GA
Number of Crews	38	31	30	31	31
Average Rotation Time	216.658	225.258	226.533	213.903	211.323
STD of Rotation Times	52.554	63.79	63.348	69.652	68.496
Number of Dead-headings	50	26	21	19	17
Not ends where it started	0	0	0	0	0
Total Flying Time (mins)	8233	6983	6796	6631	6551
Cost	11400	9444	9120	9432	9452
Running Time (sec)	< 1	1	536	2	7826

Table 15: Results for GBXDFW_M_86 problem (86 flights). The sum of flying times is 5521 minutes. CBGA is run for 1000 generations. Graph based Steady State GA is run 5000 iterations, population size is 20, mutation probability is 0.4, crossover probability 0.5.

	Chavatal [17] SCP	Back [18] SCP	SCP based [19] CBGA	Greedy#3 SCP	The Best with GA
Number of Crews	28	23	23	23	23
Average Rotation Time	218.893	236.261	236.261	227.565	227.565
STD of Rotation Times	56.691	70.085	70.222	72.153	72.801
Number of Dead-headings	29	16	16	12	12
Not ends where it started	0	0	0	0	0
Total Flying Time (mins)	6129	5434	5434	5234	5234
Cost	8400	7054	7080	7107	7092
Running Time (sec)	< 1	1	371	1	3267

Table 16: Results for GBXDFW_M_63 problem (63 flights). The sum of flying times is 4389 minutes. CBGA is run for 1000 generations. Graph based Steady State GA is run 2000 iterations, population size is 20, mutation probability is 0.4, crossover probability 0.5.

	Chavatal [17] SCP	Back [18] SCP	SCP based [19] CBGA	Greedy#3 SCP	The Best with GA
Number of Crews	42	33	31	33	32
Average Rotation Time	234.643	241.818	241.452	208.788	201.719
STD of Rotation Times	59.215	67.204	68.629	88.472	88.966
Number of Dead-headings	67	37	30	21	16
Not ends where it started	0	0	0	0	0
Total Flying Time (mins)	9855	7980	7485	6890	6455
Cost	12940	10310	9680	10310	10100
Running Time (sec)	< 1	< 1	560	2	2437

Table 17: Results for GBXSEA_M_75 problem (75 flights). The sum of flying times is 5420 minutes. CBGA is run for 1000 generations. Graph based Steady State GA is run 2000 iterations, population size is 20, mutation probability is 0.4, crossover probability 0.5.

	Chavatal [17] SCP	Back [18] SCP	SCP based [19] CBGA	Greedy#3 SCP	The Best with GA
Number of Crews	36	36	35	36	34
Average Rotation Time	347.111	344.25	353.543	344.25	351.441
STD of Rotation Times	109.119	109.217	99.628	109.217	105.243
Number of Dead-headings	14	12	12	12	12
Not ends where it started	0	0	0	0	0
Total Flying Time (mins)	12496	12393	12374	12393	11949
Cost	16070	15903	15598	15903	15576
Running Time (sec)	< 1	1	1027	1	2469

Table 18: Results for B737_May1_85 problem (85 flights). The sum of flying times is 11075 minutes. CBGA is run for 1000 generations. Graph based Steady State GA is run 400 iterations, population size is 20, mutation probability is 0.4, crossover probability 0.5.

	Chavatal [17] SCP	Back [18] SCP	SCP based [19] CBGA	Greedy#3 SCP	The Best with GA
Number of Crews	5	4	5	4	4
Average Rotation Time	902	1127.5	902	1127.5	1127.5
STD of Rotation Times	152.204	295.434	246.609	443.572	182.671
Number of Dead-headings	0	0	0	0	0
Not ends where it started	0	0	0	0	0
Total Flying Time (mins)	4510	4510	4510	4510	4510
Cost	10441.5	9088.5	9756.5	9118.5	8349.5
Running Time (sec)	< 1	< 1	153	< 1	697

Table 19: Results for LynxAir problem (32 flights). The sum of flying times is 4510 minutes. CBGA is run for 1000 generations. Graph based Steady State GA is run 400 iterations, population size is 20, mutation probability is 0.4, crossover probability 0.5.

	Chavatal [17] SCP	Back [18] SCP	SCP based [19] CBGA	Greedy#3 SCP	The Best with GA
Number of Crews	23	17	16	17	16
Average Rotation Time	277.174	339.706	354.688	310.294	329.688
STD of Rotation Times	78.169	118.278	93.732	151.623	87.329
Number of Dead-headings	23	11	9	1	1
Not ends where it started	0	0	0	0	0
Total Flying Time (mins)	6375	5775	5675	5275	5275
Cost	14984	11414	10784	11530	10678
Running Time (sec)	1	6	752	15	972

Table 20: Results for GBXORD_MT_73 problem (73 flights). The sum of flying times is 5225 minutes. CBGA is run for 1000 generations. Graph based Steady State GA is run 1000 iterations, population size is 20, mutation probability is 0.4, crossover probability 0.5.

	Chavatal [17] SCP	Back [18] SCP	SCP based [19] CBGA	Greedy#3 SCP	The Best with GA
Number of Crews	27	23	23	23	22
Average Rotation Time	298.333	301.957	298.478	298.478	312.045
STD of Rotation Times	104.978	108.033	89.533	109.32	78.5473
Number of Dead-headings	20	4	2	2	2
Not ends where it started	0	0	0	0	0
Total Flying Time (mins)	8055	6945	6865	6865	6865
Cost	10050	8750	8290	8750	8180
Running Time (sec)	< 1	1	634	3	997

Table 21: Results for GBXDEN_M_72 problem (72 flights). The sum of flying times is 6660 minutes. CBGA is run for 1000 generations. Graph based Steady State GA is run 1000 iterations, population size is 20, mutation probability is 0.4, crossover probability 0.5.

	Chavatal [17] SCP	Back [18] SCP	SCP based [19] CBGA	Greedy#3 SCP	The Best with GA
Number of Crews	38	21	20	21	18
Average Rotation Time	225.395	269.762	267.25	238.333	263.611
STD of Rotation Times	55.421	36.399	59.065	66.068	67.614
Number of Dead-headings	92	22	16	6	2
Not ends where it started	0	0	0	0	0
Total Flying Time (mins)	8565	5665	5345	5005	4745
Cost	11400	6360	6380	6360	5800
Running Time (sec)	1	6	896	10	3071

Table 22: Results for GBXBOS_M_73 problem (73 flights). The sum of flying times is 4600 minutes. CBGA is run for 1000 generations. Graph based Steady State GA is run 400 iterations, population size is 20, mutation probability is 0.4, crossover probability 0.5.

	Chavatal [17] SCP	Back [18] SCP	SCP based [19] CBGA	Greedy#3 SCP	The Best with GA
Number of Crews	46	29	32	28	29
Average Rotation Time	244.348	305.517	279.062	312.5	301.724
STD of Rotation Times	68.386	39.791	60.018	48.962	35.043
Number of Dead-headings	42	2	4	0	0
Not ends where it started	0	0	0	0	0
Total Flying Time (mins)	11240	8860	8930	8750	8750
Cost	14114	9755	10679	9740	9580
Running Time (sec)	< 1	4	1561	10	3684

Table 23: Results for GBXATL_M_118 problem (118 flights). The sum of flying times is 8750 minutes. CBGA is run for 1000 generations. Graph based Steady State GA is run 2000 iterations, population size is 20, mutation probability is 0.4, crossover probability 0.5.

	Chavatal [17] SCP	Back [18] SCP	SCP based [19] CBGA	Greedy#3 SCP	The Best with GA
Number of Crews	46	36	37	36	35
Average Rotation Time	190.478	222.278	218.162	219.222	225.486
STD of Rotation Times	64.984	74.049	63.917	73.459	68.667
Number of Dead-headings	37	13	17	11	11
Not ends where it started	0	0	0	0	0
Total Flying Time (mins)	8762	8002	8072	7892	7892
Cost	14752.8	12072.8	12100.8	12072.8	11707.8
Running Time (sec)	< 1	1	966	2	3793

Table 24: Results for GBXMIA_M_93 problem (93 flights). The sum of flying times is 6963 minutes. CBGA is run for 1000 generations. Graph based Steady State GA is run 400 iterations, population size is 20, mutation probability is 0.4, crossover probability 0.5.

	Chavatal [17] SCP	Back [18] SCP	SCP based [19] CBGA	Greedy#3 SCP	The Best with GA
Number of Crews	63	44	45	45	45
Average Rotation Time	239.206	252.386	239.222	232.667	219.778
STD of Rotation Times	60.225	64.908	70.804	77.471	92.586
Number of Dead-headings	103	42	38	31	23
Not ends where it started	0	0	0	0	0
Total Flying Time (mins)	15070	11105	10765	10470	9890
Cost	19270	13680	14070	14010	14500
Running Time (sec)	< 1	2	1633	6	4476

Table 25: Results for GBXSEA_M_110 problem (110 flights). The sum of flying times is 8420 minutes. CBGA is run for 1000 generations. Graph based Steady State GA is run 2000 iterations, population size is 20, mutation probability is 0.4, crossover probability 0.5.

	Chavatal [17] SCP	Back [18] SCP	SCP based [19] CBGA	Greedy#3 SCP	The Best with GA
Number of Crews	76	71	64	71	62
Average Rotation Time	400.974	412.592	420.391	409.028	448.823
STD of Rotation Times	100.391	105.721	92.501	108.487	101.23
Number of Dead-headings	77	64	40	61	46
Not ends where it started	0	0	0	0	0
Total Flying Time (mins)	30474	29294	26905	29041	27827
Cost	39279.7	38020.5	35000.7	37696.5	37509.7
Running Time (sec)	1	4	3246	10	14042

Table 26: Results for MMMD88_May1_173 problem (173 flights). The sum of flying times is 22291 minutes. CBGA is run for 1000 generations. Graph based Steady State GA is run 2000 iterations, population size is 20, mutation probability is 0.4, crossover probability 0.5.

	Chavatal [17] SCP	Back [18] SCP	SCP based [19] CBGA	Greedy#3 SCP	The Best with GA
Number of Crews	25	22	22	22	22
Average Rotation Time	657.48	666.636	631.545	645	714.273
STD of Rotation Times	140.444	117.079	132.496	131.629	122.026
Number of Dead-headings	31	21	14	16	22
Not ends where it started	0	0	0	0	0
Total Flying Time (mins)	16337	14666	13894	14190	15714
Cost	20248.9	18744.5	16946	18440.5	20732.3
Running Time (sec)	1	8	1224	22	19054

Table 27: Results for B727_May1_86 problem (86 flights). The sum of flying times is 11658 minutes. CBGA is run for 1000 generations. Graph based Steady State GA is run 1000 iterations, population size is 20, mutation probability is 0.4, crossover probability 0.5.

	Chavatal [17] SCP	Back [18] SCP	SCP based [19] CBGA	Greedy#3 SCP	The Best with GA
Number of Crews	19	15	17	15	17
Average Rotation Time	317.632	393.667	343.235	374.333	325
STD of Rotation Times	178.631	196.108	191.003	208.278	166.035
Number of Dead-headings	27	14	13	10	9
Not ends where it started	0	0	0	0	0
Total Flying Time (mins)	6035	5905	5835	5615	5525
Cost	22042.8	18207.3	19356.5	17593.5	16914.2
Running Time (sec)	6	15	2695	46	55830

Table 28: Results for OrcaAir problem (83 flights). The sum of flying times is 4995 minutes. CBGA is run for 1000 generations. Graph based Steady State GA is run 5000 iterations, population size is 20, mutation probability is 0.4, crossover probability 0.5.

	Chavatal [17] SCP	Back [18] SCP	SCP based [19] CBGA	Greedy#3 SCP	The Best with GA
Number of Crews	163	147	139	146	133
Average Rotation Time	379.436	383.313	425.46	381.856	419.865
STD of Rotation Times	108.51	104.481	108.621	108.178	123.016
Number of Dead-headings	206	136	181	127	158
Not ends where it started	0	0	0	0	0
Total Flying Time (mins)	61848	56347	59139	55751	55842
Cost	78343.8	71984.7	78421	71714.7	77440.8
Running Time (sec)	2	52	10616	99	27934

Table 29: Results for MMMD88_May1_333 problem (333 flights). The sum of flying times is 43063 minutes. CBGA is run for 1000 generations. Graph based Steady State GA is run 400 iterations, population size is 20, mutation probability is 0.4, crossover probability 0.5.

	Chavatal [17] SCP	Back [18] SCP	SCP based [19] CBGA	Greedy#3 SCP	The Best with GA
Number of Crews	191	166	157	165	155
Average Rotation Time	373.482	390.602	428.013	388.812	433.374
STD of Rotation Times	106.013	104.79	98.724	107.665	114.16
Number of Dead-headings	244	159	193	149	188
Not ends where it started	0	0	0	0	0
Total Flying Time (mins)	71335	64840	67198	64154	67173
Cost	89482.1	82463.9	88907.5	82121.7	90517
Running Time (sec)	5	100	16427	244	12663

Table 30: Results for MMMD88_May1_380 problem (380 flights). The sum of flying times is 49295 minutes. CBGA is run for 1000 generations. Graph based Steady State GA is run 400 iterations, population size is 20, mutation probability is 0.4, crossover probability 0.5.

	Chavatal [17] SCP	Back [18] SCP	SCP based [19] CBGA	Greedy#3 SCP	The Best with GA
Number of Crews	27	18	19	18	18
Average Rotation Time	270.741	292.778	283.684	266.111	266.111
STD of Rotation Times	76.638	99.933	106.337	120.929	86.483
Number of Dead-headings	47	13	15	5	5
Not ends where it started	0	0	0	0	0
Total Flying Time (mins)	7310	5270	5390	4790	4790
Cost	16593.5	11298.5	11473.5	11319.5	11298.5
Running Time (sec)	72	127	9446	231	4948

Table 31: Results for GBXSFO_MT_97 problem (97 flights). The sum of flying times is 4560 minutes. CBGA is run for 4000 iterations. Graph based Steady State GA is run 400 iterations, population size is 20, mutation probability is 0.4, crossover probability 0.5.

	Chavatal [17] SCP	Back [18] SCP	SCP based [19] CBGA	Greedy#3 SCP	The Best with GA
Number of Crews	47	38	39	38	50
Average Rotation Time	364.255	396.447	391.872	340.474	288.7
STD of Rotation Times	99.251	131.774	102.735	152.148	84.658
Number of Dead-headings	93	60	63	28	70
Not ends where it started	0	0	0	0	0
Total Flying Time (mins)	17120	15065	15283	12938	14435
Cost	27444.2	23184	23854.8	24327	30026.2
Running Time (sec)	38	110	9356	313	8162

Table 32: Results for GBXDFW_MT_158 problem (158 flights). The sum of flying times is 10903 minutes. CBGA is run for 1000 generations. Graph based Steady State GA is run 400 iterations, population size is 20, mutation probability is 0.4, crossover probability 0.5.

Problem Name	Number of Flights	Number of columns	Number of days
GBXORD_M_26	26	16	1
Delta_B727_43	43	40	1
MandarinAir	28	52	7
GBXSEA_M_55	55	76	1
OrlandoAir	54	81	1
GBXDFW_M_63	63	107	1
GBXSFO_M_46	46	122	2
Delta_B737_85	85	123	2
LynxAir	32	133	7
GBXDFW_M_86	86	182	1
GBXSEA_M_75	75	184	1
GBXMIA_M_93	93	205	1
GBXDEN_M_72	72	295	1
Delta_MD88_173	173	399	2
GBXSEA_M_110	110	453	1
Delta_102	102	857	1
GBXATL_M_118	118	904	1
GBXORD_MT_73	73	1008	2
GBXBOS_M_73	73	1105	1
Delta_B727_86	86	1466	2
Delta_133	133	1698	1
Delta_MD88_333	333	3428	2
Delta_MD88_380	380	4415	2
OrcaAir	83	5318	7
Delta_217	217	5618	2
GBXSFO_MT_97	97	15250	2
GBXDFW_MT_158	158	14254	2
THY_74	74	21308	7

Table 33: The characteristic of each problem. The number of columns represents the enumerated columns respecting all the constraints. These columns define the corresponding Set Cover Problem. The last column in the table gives the time span of each scheduling problem.

Problem Name	Chavatal [17] SCP	Back [18] SCP	SCP based [19] CBGA	Greedy#3 SCP	The Best with GA
GBXORD_M_26	**3600**	**3600**	**3600**	**3600**	**3600**
Delta_B727_43	9698	9730	9356	9730	**8912**
MandarinAir	22609.7	17609	18546.3	17609	**15992.1**
GBXSEA_M_55	8340	7540	7120	7553	**7112**
OrlandoAir	9636.6	8914.1	**8790.3**	8914.1	8975.33
GBXDFW_M_63	8400	**7054**	7080	7107	7092
GBXSFO_M_46	6900	**4500**	**4500**	4527	**4500**
Delta_B737_85	16070	15903	15598	15903	**15576**
LynxAir	10441.5	9088.5	9756.5	9118.5	**8349.5**
GBXDFW_M_86	11400	9444	**9120**	9432	9452
GBXSEA_M_75	12940	10310	**9680**	10310	10100
GBXMIA_M_93	14752.8	12072.8	12100.8	12072.8	**11707.8**
GBXDEN_M_72	10050	8750	8290	8750	**8180**
Delta_MD88_173	39279.7	38020.5	**35000.7**	37696.5	39435.5
GBXSEA_M_110	19270	**13680**	14070	14010	14500
Delta_102	16840	13272	12592	13310	**12210**
GBXATL_M_118	14114	9755	10679	9740	**9580**
GBXORD_MT_73	14984	11414	10784	11530	**10678**
GBXBOS_M_73	11400	6360	6380	6360	**5800**
Delta_B727_86	20248.9	18744.5	**16946**	18440.5	20732.3
Delta_133	23806.9	20335.9	20043.1	20365.8	**18961.3**
Delta_MD88_333	78343.8	71984.7	78421	**71714.7**	77440.8
Delta_MD88_380	89482.1	82463.9	88907.5	**82121.7**	90517
OrcaAir	22042.8	18207.3	19356.5	17593.5	**16914.2**
Delta_217	40398.6	**29178.6**	33340.6	29224.6	30006.6
GBXSFO_MT_97	16593.5	**11298.5**	11473.5	11319.5	**11298.5**
GBXDFW_MT_158	27444.2	**23184**	23854.8	24327	30026.2
THY_74	22636.8	15431.9	17085	19420.9	**12321.5**

Table 34: The cost of solution schedule obtained from various algorithms. The best solution in the last population is used for GA based algorithms.

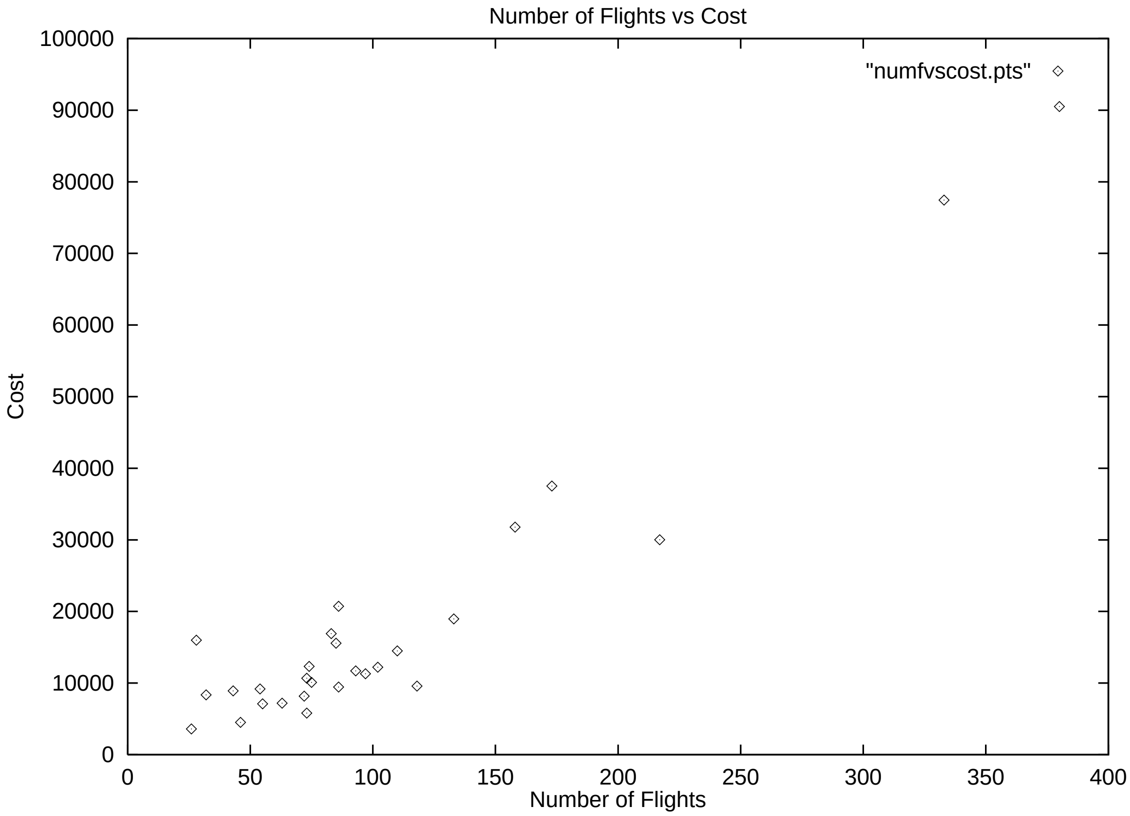

Figure 15: Cost vs number of flights

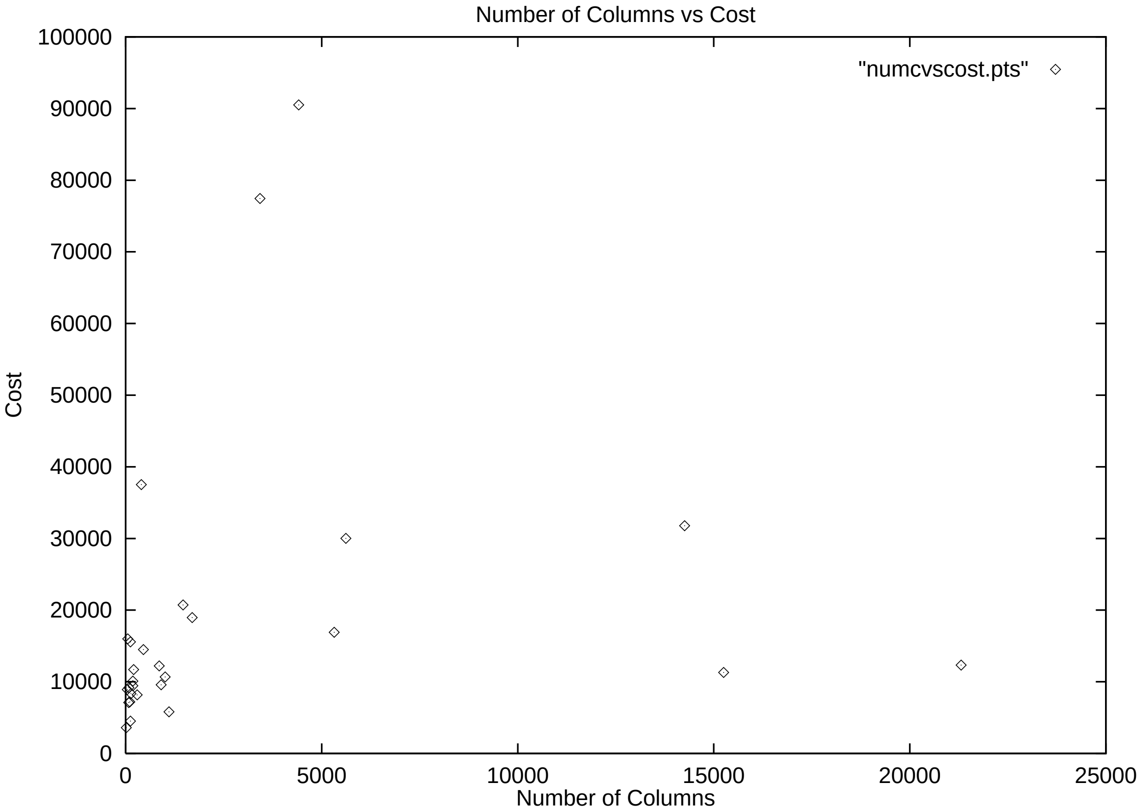

Figure 16: Cost vs number of columns