NATIONAL KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

UNDERGRADUATE THESIS

Airline Crew Pairing Problem and Meta-heuristics

Nikolaos S. Karalis

Supervisor: Panagiotis Stamatopoulos, Assistant Professor

Athens

JULY 2016

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Το πρόβλημα της δημιουργίας ανώνυμων συνδυασμών πτήσεων αεροπορικών εταιρειών και μετα-ευρετικοί αλγόριθμοι

Νικόλαος Σ. Καράλης

Επιβλέπων: Παναγιώτης Σταματόπουλος, Επίκουρος Καθηγητής

ΑΘΗΝΑ

ΙΟΥΛΙΟΣ 2016

UNDERGRADUATE THESIS

Airline Crew Pairing Problem and Meta-heuristics

Nikolaos S. Karalis

R.N.: 1115201200052

SUPERVISOR: Panagiotis Stamatopoulos, Assistant Professor

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Το πρόβλημα της δημιουργίας ανώνυμων συνδυασμών πτήσεων αεροπορικών εταιρειών και μετα-ευρετικοί αλγόριθμοι

Νικόλαος Σ. Καράλης

Α.Μ.: 1115201200052

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Παναγιώτης Σταματόπουλος,** Επίκουρος Καθηγητής

ABSTRACT

The airline crew pairing is a problem of great economical importance to the airline companies. In this thesis the most common meta-heuristics that are used to solve the problem are presented. Moreover, different genetic algorithms were implemented in order to solve the problem. The implementations were tested on the data sets of Beasley and Chu [5,11].

SUBJECT AREA: Artificial Intelligence

KEYWORDS: Airline Crew Pairing Problem, Genetic Algorithms , Meta-heuristics

ΠΕΡΙΛΗΨΗ

Η δημιουργία ανώνυμων συνδυασμών πτήσεων είναι ένα πρόβλημα μεγάλης οικονομικής σημασίας για τις αεροπορικές εταιρείες. Σε αυτή την εργασία, παρουσιάζονται οι πιο διαδεδομένοι μετα-ευρετικοί αλγορίθμοι που έχουν χρησιμοποιηθεί για να λύσουν το πρόβλημα αυτό. Επίσης, υλοποιήθηκαν διαφορετικοί γενετικοί αλγόριθμοι με σκοπό να λύσουν το πρόβλημα. Οι υλοποιήσεις αυτές εξετάστηκαν πάνω στα δεδομένα ελέγχου των Beasley και Chu [5,11].

Θεματική Ενότητα: Τεχνητή Νοημοσύνη

Λέξεις Κλειδιά: Το πρόβλημα της δημιουργίας ανώνυμων συνδυασμών πτήσεων αεροπορικών εταιρειών, Γενετικοί Αλγόριθμοι, Μετα-ευρετικοί αλγόριθμοι.

To my parents and brother.

ACKNOWLEDGMENTS

I would like to thank my supervisor, Prof. Panagiotis Stamatopoulos for giving me the opportunity to work on this subject and for helping me to carry out this work.

Contents

LIST OF FIGURES

LIST OF TABLES

1. Introduction

The airline crew pairing problem has been extensively studied due to its major economic significance. Given a set of flights, the airline companies must optimally assign their crew members to a number of flights, without leaving any of them uncovered. In other words, the pairings, that cover all the flights and minimize the cost, must be found. A pairing is a combination of flights.

The two most common representations of the airline crew pairing problem are the Set Partitioning Problem (SPP):

$$\begin{aligned} &\textit{Minimize} && \textstyle\sum_{j=1}^{n} c_j x_j \\ &\textit{Subject to} && \textstyle\sum_{j=1}^{n} a_{ij} x_j = 1, \quad i = (1, \dots, m) \\ & && x_j \in \{0,1\}, \qquad j = (1, \dots, n) \end{aligned}$$

and the Set Covering Problem (SCP):

$$\begin{aligned} &\textit{Minimize} && \textstyle\sum_{j=1}^{n} c_j x_j \\ &\textit{Subject to} && \textstyle\sum_{j=1}^{n} a_{ij} x_j \geq 1, \quad i = (1, \dots, m) \\ & && x_j \in \{0,1\}, \qquad j = (1, \dots, n) \end{aligned}$$

where c_j is the cost of pairing j and $a_{ij} \in \{0,1\}$. If $a_{ij} = 1$ then the flight i is covered by the pairing j. The columns represent the pairings, whereas the rows represent the flights. The difference between the two formulations lies in the constraints. If the airline crew pairing problem is formulated as an SPP, all flights must be covered by exactly one pairing, whereas in the case of the SCP, all flights must be covered by at least one pairing.

Since both the SPP and the SCP are NP-Compete, heuristic algorithms are used in order to solve them. Although heuristics do not guarantee optimal solutions, such algorithms are vastly used to solve problems of this nature, due to the fact that they manage to achieve optimal or near optimal solutions in feasible time. In this work the focus is set on meta-heuristics and especially on Genetic Algorithms. Meta-heuristics are algorithms that are nature-inspired [1]. We present the most common meta-heuristics and the implementation of three different genetic algorithms.

2. Related Work

In this section studies, which approached the problem using meta-heuristic algorithms, are presented.

Meta-heuristic algorithms can be divided into two categories; those that are based on a single solution(Single-Solution Based meta-heuristics) and those that are based on multiple solutions (Population based meta-heuristics). The second category is as well divided into two subcategories; the Evolutionary Algorithms (EA) and the Swarm Intelligence (SI) Algorithms.

On one hand, Evolutionary Algorithms (e.g. Genetic Algorithms, Evolution Strategy) are based on the principles of Charles Darwin. These principles suggest that living organisms evolve and adapt to their environment. Fig. 1 presents an overview of EAs.

1. Initialize the population randomly

2. Evaluate solutions

3. Select parents

4. Recombine parents

5. Mutate offspring

6. Evaluate new solution

7. Choose new solution that is going to enter the population

8. Iterate steps 3-7, until the terminating condition is met

9. Return the best solution that was found

Figure 1: Evolutionary Algorithm

On the other hand, Swarm Intelligence algorithms simulate the behavior of social insect colonies and other animal societies. Examples of such algorithms are the Ant Colony Optimization (ACO), the Particle Swarm Optimization (PSO) and others. For more detailed information regarding the meta-heuristics we refer the reader to [1]. Population-based meta-heuristics are chosen to solve the airline crew pairing problem, due to its nature.

2.1 Swarm Intelligence

2.1.1 Ant Colony Optimization

Lessing et al. in [13] compare several ACO algorithms on instances of the Set Covering Problem. These algorithms are the Min-Max Ant System (MMAS), the Ant Colony System (ACS), a hybrid algorithm combining MMAS and ACS (MMAS-ACS-Hybrid) and the Approximate Nondeterministic Tree Search (ANTS). These variations of the ACO algorithm differ on the construction of the solutions and the way the pheromone trails are updated. A local search heuristic, which is based on the *r*-Flip neighborhood, is used. The local search heuristic creates neighbor solutions by changing the value of at most *r* variables (i.e. their Hamming-Distance is at most *r*). Lessing et al. experimented with several types (static and dynamic) of heuristic information as well. The static heuristic information (e.g. Column Costs) is computed once and the same value is used in every iteration of the algorithm, whereas the dynamic heuristic information (e.g. Lagrangean Cover Costs, Cover Costs, Lower Bounds) is computed in every construction step. The results of the study show that when the local search heuristic is not used the best solutions

are obtained by ACS. ANTS combined with the Lower Bound heuristic information has the best performance, when the local search heuristic is used.

In [2] Crawford et al. (2006) solve the airline crew pairing problem, which is formulated as an SPP, in two different ways. The first one is a plain ACO (two different instances of ACO; AS and ACS) algorithm and the second is a hybrid algorithm which combines elements of Constraint Programming (CP) with the ACO algorithm. To be more specific CP is used at the stage of ACO, where the variables are selected (i.e. columns are added to the solution). Each pairing (i.e. column of the SPP) is associated with two values; the pheromone trail and a heuristic information. The pheromone trail on each pairing is related to its frequency in the ants' solutions. The heuristic operation measures the unit cost of covering one additional row. A column is added to the partial solution probabilistically. The probability depends on the pheromone trail and the heuristic information of each column. In this study the Transition Rule Probability is used as the decision policy. Forward Checking with Backtracking is the CP algorithm that is combined with the ACO. Both of these algorithms were tested on real problems of an American airline company and the computational results show that, the hybrid algorithm reaches better solutions than the plain ACO.

Ren et al. (2010) propose an ACO algorithm for the Set Covering Problem [12]. Like previous studies, each column of the problem is given a pheromone value and a heuristic value. Although, a new method is introduced for constructing the solutions. Instead of adding columns to the solution until all rows are covered, at each construction step an uncovered row is chosen. Then, a column that covers this row is selected, with probability that is based on the pheromone and heuristic values. This method is called single-row-oriented solution construction and it generates solutions faster than the method mentioned in the previous studies. The dynamic heuristic information used in this study takes into account Lagrangian dual information associated with currently uncovered rows. After each iteration the pheromones are uniformly reduced and the ants deposit pheromones on the columns that are contained in the best global solution. Moreover, a local search heuristic is developed, that aims to remove redundant columns from the solution. The computational results show that the proposed algorithm can produce good solutions and that it can compete with other meta-heuristics.

Deng and Lin (2011) in [4] formulate the problem as a Traveling Salesman Problem (TSP), that uses a flight graph representation, and propose an ACO algorithm to solve it. The objective of ACO is to find the path with minimum cost in a graph, whose nodes and edges respectively represent the flights and constraints between two consecutive flights. Each edge of the graph, is associated with a pheromone value and heuristic value. The state transition probability of the ACO is based on these values. The heuristic value is inversely proportional to the interval time of each flight (i.e. edge of the graph). After each iteration, the value of all pheromones are reduced and only the globally best ants deposit pheromones. This way, the best solutions have higher value of pheromone. The proposed algorithm is compared with a Genetic Algorithm on real cases and leads to better results for the majority of them.

Shihabi et al. (2015) present in [14] a hybrid, MMAS-based ACO algorithm for the Set Covering Problem. Before the main loop of the algorithm begins, they solve the LP-relaxation of the problem in order to reduce its size. Unlike a typical MMAS algorithm, when the pheromone values are updated, they are not checked against a lower bound. Instead they are only checked against an upper bound. The heuristic information is based on the dynamic reduced costs and the number of rows that can be covered by selecting a certain column (similar to [12]). Unlike the heuristic of [12], the normalization of the heuristic values is based on their minimum and maximum values and not on another parameter. The results show that the reduction of the size of the problem has a great

impact on the quality of the solutions and on the computation time. Fig. 2 presents an overview of ACO.

1. Initialize pheromone trails

2. Construct solutions

3. Update pheromone

4. Iterate steps 1-3, until terminating condition is met

5. Return the best solution that was found

Figure 2: Ant Colony Optimization

2.1.2 Particle Swarm Optimization

In [3] a PSO algorithm is developed for the integrated airline crew scheduling problem. The particles are associated with two vectors; their position and their velocity. The position of a particle represents a solution of the problem. In every iteration of the algorithm the position and the velocity are updated. The update of the velocity of a particle is based on the particle's best previous position and on the global best previous solution. The updated velocity determines the new position of the particle.

1. Initialize the position and the velocity of the particles randomly

2. Evaluate the position and the velocity of each particle

3. Initialize each particle's best position and the swarm's best position

4. Update each particle's velocity and position

5. Update each particle's best position and the swarm's best position

6. Iterate steps 4 and 5 until the terminating condition is met.

7. Return the best solution that was found

Figure 3: Particle Swarm Optimization

2.1.3 Artificial Bee Colony

An Artificial Bee Colony (ABC) optimization algorithm is proposed by Sundar and Singh (2012) in [15] for the Set Covering Problem. At the initialization step of the algorithm, each employed bee is associated with a randomly generated feasible solution. Onlookers choose a food source probabilistically. The roulette wheel selection method is used. The employed bees as well as the onlookers find new food sources (i.e. solutions) in their neighborhood. Firstly, they add a number of columns, that are part of randomly generated solution j, to their solution i. In case there is not a collision (i.e. solution i is not equal to solution j), a number of columns are randomly dropped from the solution. After this step, a feasibility operator is used to repair illegal solutions. If an employed bee encounters a collision, it becomes a scout. That means it is assigned a new random food source. If the solution i of an onlooker equals to the random solution j, a new solution j is randomly generated, until there is no collision. A local search heuristic, that is similar to the one proposed in [12], is used. The heuristic is applied repeatedly until there cannot be any further improvement to the solution, that it is applied on. Moreover, each food source is associated with two fitness functions. The first one (primary) is the same as the objective function, whereas the second (secondary) is equal to the number of rows that are covered by a single column. The secondary fitness function is used in case there are solutions,

whose objective functions have the same value. Lastly, an employed bee will become a scout, if its food source does not improve after a number of iterations.

1. Initialization

2. Work of employed bees

3. Work of onlooker bees

4. Work of scout bees

5. Save the best solution

6. Iterate steps 2 - 5 until the terminating condition is met.

7. Return the best solution that was found

Figure 4: Artificial Bee Colony

2.2 Genetic Algorithms

Now we set our focus onto studies that have solved the airline crew pairing problem using Genetic Algorithms (GA).

2.1.1 Genetic Algorithms and Set Partitioning Problem

Chu and Beasley [5] (1995) developed a steady-state GA combined with a heuristic feasibility. More specifically, the fitness of each solution is divided into two parts, the fitness and the unfitness. The fitness serves as the objective function of the problem and the unfitness measures the infeasibility of each individual. Moreover, for the selection of the parents, a method called Maximum Compatibility Selection (MCS) is developed. Its goal is to improve the fitness of the solutions and to make them more feasible. Concerning the goal of MCS, it is essential that the selected parents cover as much columns of the Set Partitioning Problem as possible and at the same time keep their common rows at low numbers. The Subgroup Ordering Replacement (SOR) scheme is created for the replacement stage of the algorithm. The SOR scheme splits the individuals of the population into four subgroups. The splitting is based on the offspring, which is about to enter the population and the current solutions are assigned to their corresponding subgroup based on the values of their fitness and unfitness. SOR aims to first get rid of the solutions with high unfitness and then of the solutions with high fitness. The feasibility operator consists of two stages. In the first one, it finds over-covered rows and randomly removes columns from them in order to be covered by, at most, one column. In the second stage, it finds the under-covered rows and randomly adds columns, without over-covering any of them. The presented GA was tested on 55 real cases and wasn't able to find a feasible solution for only four of them.

Levine (1996) in [6] approaches the problem with a steady-state GA combined with a local search heuristic. He introduces a linear penalty term on the fitness function, whose value depends on the constraint that is violated. Binary tournament selection is used to find the parents that are going to produce the new solutions. In each iteration of the algorithm either the crossover or the mutation operator is randomly chosen. In case the mutation operator is chosen, it is applied randomly on one of the two parents that are selected. The crossover operator that is used in this study is the Two-point crossover operator that produces two new individuals. The purpose of the local search heuristic is to help the GA to find feasible solutions. The computational results confirm that its use leads to feasible solutions on most real cases that the algorithm was tested on.

Kotecha et al. (2004) [8] choose a different way to represent the solutions of the problem. The representation used is called real value encoding (row based encoding). The length of the chromosomes is equal to the number of flights (rows). The value of a flight indicates the pairing that covers this flight. In this study, the Cost-Based Uniform Crossover (CUC) operator is introduced. CUC is a modification of the Uniform Crossover operator. Genes are passed from the selected parents to the offspring based on their costs. The mutation happens implicitly during the repair mechanism which is used when the CUC produces an infeasible solution. The repair mechanism tries to find columns (pairings) that cover under-covered flights. In case a solution is not repairable it is discarded and a new one is generated. This study is compared with [6] and this comparison shows that CUC leads to better results.

Elhabashy et al.(2014) combine the works of previous studies in [10]. They experiment with two different crossover operators. The first one is the single-point crossover that produces two new solutions. The second one is the fusion crossover proposed in [11]. The single-point crossover operator is used to increase the diversity of the population. Three different mutation operators are implemented. The first one is suggested by Levine ([6]). The second is the dynamic mutation operator that is proposed in [7]. The Perturbation Operator from [9] is the last one.

2.2.2 Genetic Algorithms and Set Covering Problem

Beasley and Chu (1996) use a steady-state GA to solve the Set Covering Problem ([11]). The fusion crossover operator is proposed. If the first parent chromosome is fitter than the second, then the genes of the first parent are more likely to be passed to the offspring. In addition, they present an equation that calculates the number of genes that are mutated in each iteration. This number depends on the convergence rate of the GA. The feasibility operator tries to repair infeasible solutions by finding columns (pairings) with low cost that cover as many uncovered rows as possible.

Kornilakis and Stamatopoulos [7] (2002) solve the problem in two stages. In the first stage a large number of pairings is produced. The second stage is the optimization, where the GA is used. Over-covered rows (dead-head flights) add a penalty to the fitness function. The selection of the parents is based on their fitness value, i.e. fitter solutions are more likely to be selected. A uniform crossover operator that produces a new individual and a dynamic mutation operator are used. The mutation operator randomly chooses a number of genes of the produced chromosome, which are then probabilistically mutated to 0 (or 1) depending on the percentage of 0s (or 1s) found in the fittest solution of the population. In case a produced individual is infeasible (i.e. it contains under-covered flights), the value of some genes are changed to 1 so that all flights are covered. The replacement method that is used in this study is elitism. This means that the least fit solution is more likely to be replaced by the offspring in the new population.

Zeren and Özkol (2012) in [9] use binary selection tournament to select the parents. The fusion crossover operator is used ([11]) to produce the child solution. In each iteration the number of genes to be mutated is determined (like in [11]). The mutation operator used is the one proposed in [7]. After the crossover and the mutation, the feasibility operator, that is proposed in [11], is used to repair illegal (under-covered) solutions. In this study a new operator, the Perturbation Operator, is introduced in the genetic iteration. The Perturbation Operator deliberately makes the new solutions infeasible. Its goal is to make the solutions feasible again by covering under-covered flights with pairings of lower cost. In case the operator fails to find such pairings, the chromosomes are reverted to their original state. Elitism is chosen to be the replacement method. The 20 fittest solutions are kept after each iteration. The results show that the GA with the Perturbation Operator returns better results in less time compared to [11].

3. Genetic Algorithms

A genetic algorithm can be understood as an "intelligent" probabilistic search algorithm, which can be applied to a variety of combinatorial optimization problems [11]. Genetic algorithms simulate the natural evolution of living organisms. Organisms live and adapt to their environment. Individuals that are successful at adapting are more likely to survive and pass their genes to the new generations, whereas individuals who fail to adapt are more likely to be eliminated. By combining the genes of fit individuals, offsprings are produced, which are probably fitter than their ancestors.

The simulation of this process begins with the random initialization of the population. Each individual of the population represents a solution to the problem. Fitter individuals are chosen for crossover, where pairs of parent solutions produce one or two offspring solutions. The produced individuals are then mutated, i.e. some of their genes are altered probabilistically. In the cases of the SPP and the SPC, some of the produced solutions are infeasible. A feasibility operator is applied on infeasible solutions, in order to repair them. At the end of each iteration of a genetic algorithm the fittest individuals are kept in the population. This replacement method is called elitism. The algorithm is terminated after a given number of iterations.

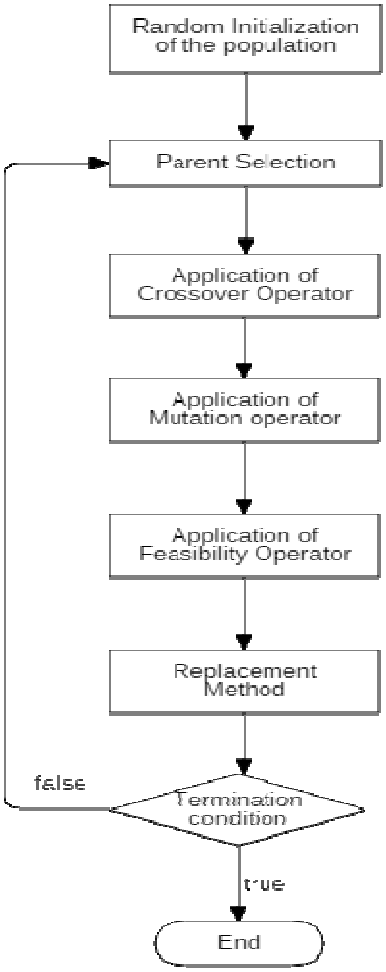

Figure 5: Genetic Algorithm

4. Genetic Algorithm Implementations

In this section we present three genetic algorithms that were implemented in order to solve the airline crew pairing problem, that was formulated not only as a Set Covering Problem but as a Set Partitioning Problem as well.

4.1 Set Covering Problem

The SCP formulation is approached by two different implementations. The most important difference between the two approaches is the crossover operator. In the first one the fusion crossover operator from [11] is used, whereas in the second approach the one-point crossover is used.

Chromosome Encoding and Objective Function

The objective function of a solution is the total cost of the pairings that belong to this solution. Each chromosome of the population represents a potential solution to the problem. Column-based encoding is selected for both approaches. Each gene of a chromosome represents a pairing. A gene takes one out of two values, 0 or 1. If the value of the i_{th} gene is equal to 1 then the i_{th} pairing is a part of the solution.

Mutation Operator

The mutation operator that is implemented in both approaches is inspired by the dynamic mutation operator that is proposed in [7]. Each gene has the same probability to be mutated. If a gene is selected for mutation and its current value is 0 (or 1) then the value of this gene is altered to 1 (or 0) with a probability that is equal to the number of genes that are equal to 1 (or 0) divided by the number of the total genes of the solution.

Feasibility Operator

After the crossover and the mutation operator the resulting solution might be infeasible. That means that some of the rows, i.e. flights, of this individual are uncovered. The feasibility operator is applied on such solutions in order to make the feasible. The feasibility operator that is implemented is greedy. For each uncovered row, the operator inserts into the solution the pairing that covers this row and has the minimum cost.

Local Search

In order to make the individuals better, a local search is implemented. Its purpose is to remove redundant pairings from the solutions. Firstly, the local search finds the flights that are covered by at least two pairings. For each over-covered row, it finds the pairings that cover this row and are part of the solution. Then it starts removing such pairings, without making the solutions infeasible.

Replacement Method

Elitism is chosen as the replacement method. After each iteration of the algorithm, the fittest individuals are kept in the population.

4.1.1 SCP with Fusion Crossover

Parent Selection

In this approach, the Binary Tournament is chosen as the parent selection method. In each iteration two parents are selected for reproduction purposes. Fitter solutions are more likely to be selected.

Crossover Operator

The fusion crossover operator is introduced by Beasley and Chu in [11]. It is a probabilistic operator and produces one new solution. The genes of the fitter parent are more likely to be passed to the offspring.

$$(i) \quad C[i] = P_1[i], if\ P_1[i] = P_2[i]$$
$$(ii)\ (a)\ C[i] = P_1[i], with\ probability\ p = \frac{f_2}{f_1 + f_2}$$
$$(b)\ C[i] = P_2[i], with\ probability\ (1-p)$$

where f_1 and f_2 are the values of the fitness function of the first and second parent respectively.

4.1.2 SCP with One-point Crossover

Parent Selection

Each individual has the same probability to be selected. At each iteration of the algorithm, the number of parents is even.

Crossover Operator

The one-point crossover produces two new solutions. The crossover point is randomly chosen. The children are produced by swapping the segments of the two parents [11].

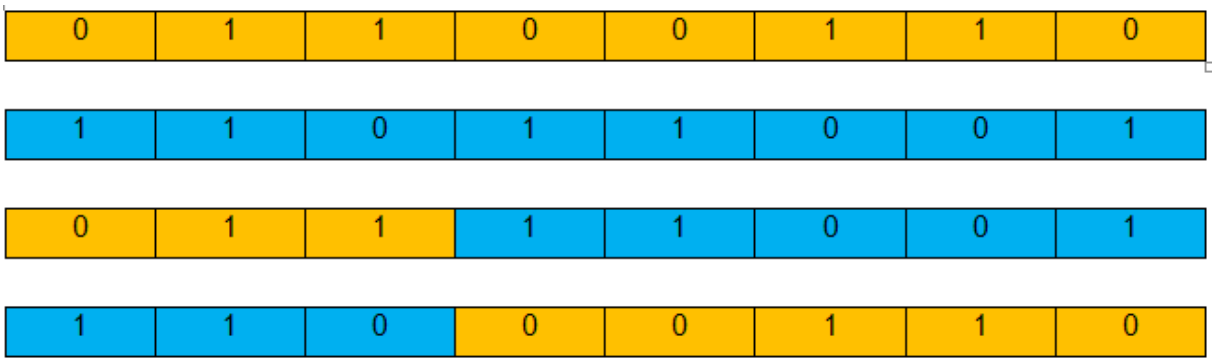

Figure 6: One-point Crossover

4.2 Set Partitioning Problem

Chromosome Encoding and Objective Function

The objective function is the same as the one used in the SCP approach. However the chromosome encoding is different. Row-encoding is used for the SPP. Each gene represents a flight. The value of each gene is an integer number in the range $[1, number\_of\_columns]$. The value of a gene indicates the pairing that is covering the flight represented by this particular gene.

Parent Selection and Crossover Operator

The parent selection method and the crossover operator are the same as those used in section 4.1.2.

Mutation Operator

Each gene has the same probability to be mutated. The new value is a pairing that also covers this flight. In this way, there is not a flight which remains uncovered.

Feasibility Operator

After the crossover and the mutation, the solutions might be infeasible, i.e. there are over-covered flights. The feasibility operator tries to make such solutions feasible by removing redundant pairings. Since the SPP is highly constrained, the operator's success is not guaranteed.

Replacement Method

Just as the SCP approach, elitism is the replacement method. Between infeasible solutions, those who have less over-covered rows are considered fitter.

5. Experimental Results

The data sets from the OR Library of Beasley [5,11] were given as input to the proposed algorithms. The algorithms were coded on Java and tested on a Intel(R) Core(TM) i3-4170 CPU @ 3.70GHz. In the tables that are presented below, the column "Problem" contains the name of each problem, the column "OPT/BKS" contains the optimal/best known solution of each problem. Due to their non-deterministic nature, the algorithms were executed several times. The columns "1st" to "5th"/"7th" are different executions of the problems and their cells contain the value of the best solution that was found in each execution. The cells of the last column, namely "AVG CPU TIME", hold the average CPU time in seconds that was needed for each problem to complete.

5.1 Set Covering Problem

The experimental results of the two different approaches for the set covering problem are presented below.

5.1.1 Set Covering Problem with Fusion Crossover

This approach was tested using four different sets of parameters on 30 problems from OR Library. The four sets of parameters are listed below.

First set:

- Population Size = 20
- Number of Iterations = 30000
- Mutation Probability p = 0.05
- Probability p' of Binary Tournament: 0.20

Second set:

- Population Size = 20
- Number of Iterations = 30000
- Mutation Probability p = 0.08
- Probability p' of Binary Tournament: 0.25

Third set:

- Population Size = 15
- Number of Iterations = 30000
- Mutation Probability p = 0.10
- Probability p' of Binary Tournament: 0.30

Fourth set:

- Population Size = 25
- Number of Iterations = 30000
- Mutation Probability p = 0.12
- Probability p' of Binary Tournament: 0.18

Table 1: SCP fusion crossover first set

Problem	OPT/BKS	1st	2nd	3rd	4th	5th	6th	7th	AVG CPU TIME
Scp41	429	443	432	432	438	442	436	432	21.0129
Scp42	512	512	540	512	512	512	533	512	18.7136
Scp43	516	528	518	516	528	517	528	516	19.0210
Scp44	494	509	512	509	512	500	529	512	20.0760
Scp45	512	518	518	518	518	518	517	517	18.5089
Scp46	560	562	568	570	562	573	560	562	19.0755
Scp47	430	433	433	432	433	433	433	433	18.2809
Scp48	492	493	499	499	499	499	497	499	19.2598
Scp49	641	660	663	655	660	655	660	655	18.8521
Scp410	514	546	543	517	544	547	547	520	19.3459
Scp51	253	260	260	264	261	253	253	260	35.6548
Scp52	302	313	306	308	308	313	308	313	30.7857
Scp53	226	230	230	230	231	231	230	226	48.3351
Scp54	242	243	242	243	243	242	243	242	42.9212
Scp55	211	212	212	211	211	212	211	211	34.1253
Scp56	213	226	227	214	237	214	232	237	25.3318
Scp57	293	301	308	307	308	308	306	313	27.4383
Scp58	288	297	291	300	302	301	308	304	26.8516
Scp59	279	280	280	280	281	283	292	291	23.6058
Scp510	265	272	265	268	273	265	271	272	28.0885
Scp61	138	143	148	147	143	143	145	145	20.4567
Scp62	146	150	151	151	150	151	149	147	19.1676
Scp63	145	145	150	148	145	150	148	145	23.5578
Scp64	131	131	131	132	132	135	132	132	19.0770
Scp65	161	164	161	161	165	161	175	161	18.6749
Scpa1	253	255	255	255	255	255	255	257	65.6941
Scpa2	252	266	265	263	258	262	262	264	74.5672
Scpa3	232	244	239	241	238	238	242	239	72.9458
Scpa4	234	239	239	244	246	239	244	238	69.5138
Scpa5	236	239	236	238	238	240	236	238	69.1541

Table 2: SCP fusion crossover second set

Problem	OPT/BKS	1st	2nd	3rd	4th	5th	6th	7th	AVG CPU TIME
Scp41	429	438	437	437	443	430	438	443	23.4884
Scp42	512	556	512	512	530	541	541	521	18.6707
Scp43	516	528	528	528	516	522	528	529	21.5846
Scp44	494	512	520	520	512	529	512	529	19.7701
Scp45	512	518	518	514	518	518	518	517	21.4739
Scp46	560	561	573	564	561	574	560	562	22.6518
Scp47	430	433	431	439	432	430	433	433	22.9302
Scp48	492	497	499	497	499	499	499	497	25.0373
Scp49	641	655	662	666	659	663	663	655	22.1025
Scp410	514	537	546	552	519	542	519	554	20.4363
Scp51	253	264	266	261	264	260	273	269	30.7530
Scp52	302	313	313	313	311	313	313	313	33.1131
Scp53	226	231	231	230	230	230	230	230	27.6089
Scp54	242	245	245	243	245	245	245	243	29.1724
Scp55	211	211	212	211	211	212	212	212	29.4056
Scp56	213	214	232	226	233	236	214	232	27.9219
Scp57	293	308	312	303	310	308	312	308	29.5610
Scp58	288	304	301	299	304	298	298	288	30.0938
Scp59	279	288	288	291	288	286	289	292	28.3607
Scp510	265	272	265	272	272	268	266	272	30.2492
Scp61	138	145	145	145	151	143	145	145	23.6128
Scp62	146	150	151	151	151	151	147	151	20.6181
Scp63	145	150	150	150	145	150	150	150	20.4167
Scp64	131	132	132	131	132	132	132	134	23.2174
Scp65	161	161	161	171	164	161	165	164	20.8500
Scpa1	253	261	255	255	255	255	260	255	82.5397
Scpa2	252	262	266	264	260	262	261	260	81.9745
Scpa3	232	242	235	237	243	236	243	242	79.6414
Scpa4	234	239	241	240	237	243	245	238	74.8716
Scpa5	236	239	239	240	238	239	239	239	73.3673

Table 3: SCP fusion crossover third set

Problem	OPT/BKS	1st	2nd	3rd	4th	5th	6th	7th	AVG CPU TIME
Scp41	429	438	432	438	432	438	438	441	21.5105
Scp42	512	533	512	518	547	512	540	540	20.4446
Scp43	516	518	520	528	516	528	528	528	21.3335
Scp44	494	509	512	503	516	511	505	498	19.4025
Scp45	512	518	518	518	518	518	517	514	19.4383
Scp46	560	585	574	564	570	586	574	560	20.7097
Scp47	430	432	433	433	433	433	438	433	18.1855
Scp48	492	499	499	497	497	499	513	499	19.3908
Scp49	641	660	661	666	661	666	653	662	21.7285
Scp410	514	550	550	535	546	518	517	558	21.9549
Scp51	253	259	263	265	268	268	263	265	31.8028
Scp52	302	308	313	308	313	311	313	313	29.7353
Scp53	226	231	230	230	234	235	231	231	27.6426
Scp54	242	243	242	242	245	243	242	243	28.1025
Scp55	211	212	212	211	211	211	211	211	25.7069
Scp56	213	236	222	214	214	217	221	217	29.8760
Scp57	293	308	312	301	304	304	310	301	32.4437
Scp58	288	301	301	306	304	298	308	302	33.1513
Scp59	279	283	280	284	289	286	281	279	28.6000
Scp510	265	265	265	268	272	272	266	267	28.9574
Scp61	138	145	138	145	145	143	149	143	26.2052
Scp62	146	151	150	151	151	151	152	151	21.5395
Scp63	145	148	150	150	150	148	150	148	21.2777
Scp64	131	132	131	132	132	135	135	135	25.1530
Scp65	161	161	161	161	164	173	177	161	22.4013
Scpa1	253	258	258	255	260	258	259	258	79.1766
Scpa2	252	260	275	279	263	262	266	281	81.4079
Scpa3	232	247	247	242	243	242	244	246	74.7733
Scpa4	234	247	240	240	240	239	244	238	77.4462
Scpa5	236	240	240	239	236	238	238	242	81.3286

Table 4: SCP fusion crossover fourth set

Problem	OPT/BKS	1st	2nd	3rd	4th	5th	6th	7th	AVG CPU TIME
Scp41	429	438	430	432	432	432	433	446	25.7125
Scp42	512	548	512	512	512	540	512	512	22.0225
Scp43	516	516	518	516	520	518	516	522	20.7871
Scp44	494	520	510	498	512	510	512	508	22.1518
Scp45	512	518	518	518	517	517	518	518	20.4968
Scp46	560	569	561	572	561	562	570	560	22.8565
Scp47	430	433	433	433	433	433	433	439	20.5028
Scp48	492	499	497	499	499	499	497	497	22.9962
Scp49	641	655	666	660	666	655	671	660	20.3098
Scp410	514	543	528	540	540	535	562	543	21.0828
Scp51	253	263	260	262	254	261	267	262	34.3798
Scp52	302	313	315	311	311	311	316	311	32.8454
Scp53	226	231	226	234	232	228	231	230	29.7204
Scp54	242	245	247	243	245	242	244	243	31.6180
Scp55	211	211	212	211	211	212	211	211	28.3998
Scp56	213	214	214	235	239	222	214	214	30.9698
Scp57	293	305	306	312	308	312	312	303	33.1998
Scp58	288	293	298	301	304	299	293	297	36.6296
Scp59	279	280	291	284	280	288	281	281	31.1161
Scp510	265	270	267	272	266	268	273	274	33.9168
Scp61	138	145	142	143	143	143	142	145	22.7366
Scp62	146	151	151	151	151	151	151	150	24.2598
Scp63	145	149	150	148	148	150	150	150	21.6757
Scp64	131	132	132	135	132	132	132	134	25.6442
Scp65	161	164	167	167	161	174	171	172	25.9891
Scpa1	253	269	280	268	258	276	265	266	87.6003
Scpa2	252	262	266	262	268	262	291	264	86.7707
Scpa3	232	243	247	243	243	240	242	243	87.2608
Scpa4	234	251	240	235	248	246	254	246	81.8247
Scpa5	236	239	240	239	239	238	240	238	82.2204

The above results show that the first set of parameters managed to achieve optimal solutions to more problems than the rest of the sets, whereas the fourth set achieved the lowest number of optimal solutions. The parameters that influence the results the most are the probabilities p and p'. If the probability of the Binary Tournament p' is low, then the impact that less fit individuals have on the population is higher. On the other hand, the mutation probability p has a negative impact on the results if it is given a relatively high value. In the fourth set, p' has the lowest value and p the highest value in comparison to the other sets.

5.1.2 Set Covering Problem with One-point Crossover

The second approach to the Set Covering Problem was tested in a similar way. The results indicate that the parent selection probability need to be relatively high in order for the algorithm to reach good solutions. The last set of parameters was the most successful. For the majority of the problems it managed to reach the optimal solution or near-optimal solutions. It should be noted that the higher the parent selection probability is, the longer it takes for the algorithm to finish. The sets of parameters and their results are presented below.

First set:

- Population Size = 20
- Number of Iterations = 3000
- Mutation Probability p = 0.05
- Parent Selection Probability p': 0.30

Second set:

- Population Size = 20
- Number of Iterations = 3500
- Mutation Probability p = 0.08
- Parent Selection Probability p': 0.40

Third set:

- Population Size = 40
- Number of Iterations = 3000
- Mutation Probability p = 0.05
- Parent Selection Probability p': 0.25

Fourth set:

- Population Size = 40
- Number of Iterations = 3000
- Mutation Probability p = 0.03
- Parent Selection Probability p': 0.40

Table 5: SCP one-point crossover first set

Problem	OPT/BKS	1st	2nd	3rd	4th	5th	6th	7th	AVG CPU TIME
Scp41	429	434	432	432	438	438	432	432	9.8181
Scp42	512	528	535	548	540	512	512	534	10.7911
Scp43	516	523	525	517	524	516	518	528	10.2120
Scp44	494	532	495	507	512	515	494	498	12.3485
Scp45	512	518	518	518	514	526	518	517	11.8877
Scp46	560	561	572	590	568	572	562	563	13.7227
Scp47	430	433	433	433	432	433	433	439	13.5010
Scp48	492	499	497	499	497	499	499	497	13.4489
Scp49	641	666	660	668	650	662	657	663	12.2621
Scp410	514	527	543	537	530	547	532	547	14.3670
Scp51	253	256	266	262	267	254	269	272	21.1917
Scp52	302	323	314	313	315	319	315	320	21.0623
Scp53	226	230	230	231	230	229	230	231	15.80478
Scp54	242	248	242	244	242	245	243	242	15.3942
Scp55	211	212	212	218	211	219	212	212	14.6613
Scp56	213	227	236	217	232	214	214	219	15.2381
Scp57	293	308	301	309	310	305	302	305	15.6532
Scp58	288	293	304	301	301	302	298	304	16.2382
Scp59	279	281	290	281	283	283	283	290	14.5375
Scp510	265	269	265	272	272	268	273	269	14.4140
Scp61	138	148	151	143	149	143	148	138	10.3082
Scp62	146	151	151	151	154	151	151	151	10.5643
Scp63	145	145	150	150	150	150	150	150	9.8433
Scp64	131	131	132	132	132	132	132	132	9.6974
Scp65	161	169	171	168	167	176	171	171	9.7880
Scpa1	253	266	262	266	260	260	259	256	35.3104
Scpa2	252	270	262	262	274	272	274	264	37.3639
Scpa3	232	253	243	246	246	247	248	245	39.3966
Scpa4	234	243	243	248	250	247	240	248	35.7380
Scpa5	236	239	241	239	240	239	240	238	36.9488

Table 6: SCP one-point crossover second set

Problem	OPT/BKS	1st	2nd	3rd	4th	5th	6th	7th	AVG CPU TIME
Scp41	429	437	432	432	432	435	432	432	17.4724
Scp42	512	518	512	512	512	512	512	533	15.9680
Scp43	516	525	518	528	518	518	516	518	15.3840
Scp44	494	512	515	501	498	508	504	523	16.5586
Scp45	512	518	518	518	514	514	517	518	15.9611
Scp46	560	578	570	561	560	571	561	561	16.9890
Scp47	430	433	439	433	433	433	433	433	17.2101
Scp48	492	495	499	499	499	497	497	499	16.0078
Scp49	641	660	660	655	661	657	667	658	16.1707
Scp410	514	545	547	526	541	550	537	538	16.1837
Scp51	253	269	266	263	269	261	254	263	24.6621
Scp52	302	319	312	317	314	312	313	308	25.8210
Scp53	226	231	231	231	231	231	228	230	23.2509
Scp54	242	243	243	244	245	245	245	242	24.9432
Scp55	211	211	212	211	212	212	211	218	24.2837
Scp56	213	214	222	217	221	234	226	231	24.6342
Scp57	293	298	304	314	293	308	308	308	26.2531
Scp58	288	296	297	302	298	297	293	293	25.4230
Scp59	279	290	289	281	288	288	281	287	24.2408
Scp510	265	271	272	274	271	272	273	270	30.2387
Scp61	138	143	145	145	145	150	143	143	19.4016
Scp62	146	152	150	150	154	151	152	151	18.4041
Scp63	145	150	150	150	150	150	150	148	19.0133
Scp64	131	132	132	132	138	132	132	132	18.4975
Scp65	161	161	171	164	163	163	172	161	17.6584
Scpa1	253	262	260	259	261	260	259	253	60.7527
Scpa2	252	272	269	272	274	269	269	290	68.4845
Scpa3	232	246	244	246	240	252	251	248	64.9335
Scpa4	234	243	239	253	247	239	237	239	63.3171
Scpa5	236	239	240	242	239	239	240	239	66.2644

Table 7: SCP one-point crossover third set

Problem	OPT/BKS	1st	2nd	3rd	4th	5th	6th	7th	AVG CPU TIME
Scp41	429	437	432	432	437	433	437	432	19.1709
Scp42	512	512	512	533	533	512	512	532	15.5740
Scp43	516	517	518	518	522	522	528	518	15.3375
Scp44	494	523	512	529	508	495	527	512	17.2258
Scp45	512	514	518	518	518	518	518	517	16.9060
Scp46	560	569	561	561	560	564	564	564	17.2015
Scp47	430	432	433	433	433	432	432	433	15.3691
Scp48	492	497	499	497	495	497	497	499	16.9255
Scp49	641	662	655	661	661	655	660	653	18.7886
Scp410	514	549	550	526	543	516	525	533	16.8945
Scp51	253	267	267	265	270	254	266	262	29.6499
Scp52	302	312	306	308	313	309	309	314	29.6340
Scp53	226	230	230	231	230	231	228	230	34.9070
Scp54	242	245	246	242	242	243	242	243	37.6054
Scp55	211	212	211	211	211	212	217	211	33.2320
Scp56	213	232	232	237	226	233	214	214	25.7412
Scp57	293	307	305	303	304	300	310	304	26.1056
Scp58	288	301	297	293	297	301	301	293	26.5918
Scp59	279	291	288	288	281	280	280	292	25.2772
Scp510	265	267	268	267	267	267	267	267	26.4362
Scp61	138	143	145	151	138	151	143	143	17.5144
Scp62	146	151	151	151	151	151	151	152	18.4807
Scp63	145	150	150	148	150	150	150	150	18.0413
Scp64	131	131	132	132	132	132	132	132	19.0449
Scp65	161	171	168	171	163	166	164	172	18.7634
Scpa1	253	255	255	266	255	260	260	255	69.9863
Scpa2	252	268	276	265	269	261	263	264	71.2212
Scpa3	232	236	243	242	246	248	246	243	69.0105
Scpa4	234	239	236	244	241	245	245	240	65.4523
Scpa5	236	240	240	240	239	240	239	238	67.2173

Table 8: SCP one-point crossover fourth set

Problem	OPT/BKS	1st	2nd	3rd	4th	5th	6th	7th	AVG CPU TIME
Scp41	429	430	432	432	432	432	432	432	37.2136
Scp42	512	512	540	532	526	512	534	512	33.0798
Scp43	516	528	518	516	516	516	522	516	34.2717
Scp44	494	502	513	510	500	497	495	520	26.0497
Scp45	512	518	518	518	518	516	518	518	24.9219
Scp46	560	561	573	560	560	562	562	569	26.3440
Scp47	430	439	433	433	432	431	433	432	23.8103
Scp48	492	497	497	499	495	497	497	499	25.3080
Scp49	641	660	661	645	663	657	655	661	26.2824
Scp410	514	520	545	534	535	534	525	526	25.8261
Scp51	253	262	259	262	260	255	262	262	43.6478
Scp52	302	306	316	313	313	308	313	318	42.1924
Scp53	226	230	230	230	230	230	232	227	39.9239
Scp54	242	243	242	243	245	242	242	243	39.9704
Scp55	211	212	211	211	211	212	211	211	41.2342
Scp56	213	214	220	217	217	214	221	226	40.8091
Scp57	293	293	301	297	293	298	300	308	42.8903
Scp58	288	300	297	298	291	298	301	304	43.1817
Scp59	279	287	279	287	280	280	280	281	36.8587
Scp510	265	267	267	265	272	265	268	268	44.9013
Scp61	138	143	145	138	143	149	143	143	29.3501
Scp62	146	151	155	151	150	150	150	150	28.0672
Scp63	145	150	150	148	150	150	150	149	28.6059
Scp64	131	132	132	132	134	131	131	132	31.4056
Scp65	161	168	166	166	161	168	161	168	29.3265
Scpa1	253	259	255	259	255	255	255	255	120.4850
Scpa2	252	267	261	262	254	266	266	258	150.1444
Scpa3	232	237	237	239	243	240	243	242	151.5062
Scpa4	234	246	242	239	246	240	238	235	122.1941
Scpa5	236	239	239	238	240	239	238	239	104.4803

The number of iterations that are needed in the second approach is lower than the first one, since it produces more solutions in each loop of the algorithm. In the first approach only one offspring is created in each iteration.

5.2 Set Partitioning Problem

The proposed algorithm managed to find a feasible solution for seven out of ten problems. For one of those problems it also reached an optimal solution. The results of the four different sets were similar and are presented below.

First set:

- Population Size = 450
- Number of Iterations = 1000
- Mutation Probability p = 0.01
- Parent Selection Probability p': 0.40

Second set:

- Population Size = 275
- Number of Iterations = 4000
- Mutation Probability p = 0.03
- Parent Selection Probability p': 0.45

Third set:

- Population Size = 320
- Number of Iterations = 3500
- Mutation Probability p = 0.02
- Parent Selection Probability p': 0.30

Fourth set:

- Population Size = 300
- Number of Iterations = 3000
- Mutation Probability p = 0.04
- Parent Selection Probability p': 0.45

Table 9: SPP first set

Problem	OPT/BKS	1st	2nd	3rd	4th	5th	AVG CPU TIME
Sppnw01	114852	NF	NF	NF	NF	NF	82.8117
Sppnw02	105444	NF	NF	NF	NF	NF	109.3052
Sppnw03	24492	36543	34713	37926	37056	38100	78.4090
Sppnw04	16862	25206	22524	24848	26848	26200	92.2241
Sppnw05	132878	NF	NF	NF	NF	NF	200.7750
Sppnw06	7810	10792	8442	9314	9540	10280	69.1156
Sppnw07	5476	7128	6084	7120	6778	6906	56.6352
Sppnw08	35894	36978	37488	37694	36682	37906	60.4090
Sppnw09	67760	74920	83592	84390	76508	82600	65.6734
Sppnw10	68271	68847	82971	73434	72921	74460	62.8007

Table 10: SPP second set

Problem	OPT/BKS	1st	2nd	3rd	4th	5th	AVG CPU TIME
Sppnw01	114852	NF	NF	NF	NF	NF	142.3004
Sppnw02	105444	NF	NF	NF	NF	NF	134.6297
Sppnw03	24492	32886	35247	35307	40008	35607	109.6284
Sppnw04	16862	26904	24418	28798	34224	25824	159.7935
Sppnw05	132878	NF	NF	NF	NF	NF	306.8929
Sppnw06	7810	10138	9604	10414	11932	9466	74.6557
Sppnw07	5476	6294	6952	6696	6072	6326	75.3364
Sppnw08	35894	36682	38132	37740	36478	36348	60.7300
Sppnw09	67760	84248	93068	83862	83652	73904	71.7349
Sppnw10	68271	74328	83847	82794	72009	73989	57.6926

Table 11: SPP third set

Problem	OPT/BKS	1st	2nd	3rd	4th	5th	AVG CPU TIME
Sppnw01	114852	NF	NF	NF	NF	NF	90.4123
Sppnw02	105444	NF	NF	NF	NF	NF	118.3216
Sppnw03	24492	33429	41142	40278	39084	39102	87.0224
Sppnw04	16862	24978	23584	NF	25410	26494	95.8669
Sppnw05	132878	NF	NF	NF	NF	NF	267.9071
Sppnw06	7810	9604	8878	9012	9590	9286	64.4284
Sppnw07	5476	6686	7086	6084	6200	5476	57.5947
Sppnw08	35894	36682	36682	36068	38766	38862	51.3768
Sppnw09	67760	83384	102326	91974	74394	82998	59.4897
Sppnw10	68271	83121	84147	80190	82635	70596	52.8919

Table 12: SPP fourth set

Problem	OPT/BKS	1st	2nd	3rd	4th	5th	AVG CPU TIME
Sppnw01	114852	NF	NF	NF	NF	NF	153.4881
Sppnw02	105444	NF	NF	NF	NF	NF	206.2682
Sppnw03	24492	36102	32142	37716	35964	37962	102.2535
Sppnw04	16862	NF	24008	37132	33744	27428	138.037
Sppnw05	132878	NF	NF	NF	NF	NF	357.8799
Sppnw06	7810	8382	12326	9170	9196	9494	78.2370
Sppnw07	5476	6996	6084	6786	6200	6312	71.0792
Sppnw08	35894	36682	37694	36578	38482	36682	56.6291
Sppnw09	67760	73160	82050	74256	83638	83262	64.332
Sppnw10	68271	73368	72933	NF	73971	72396	59.5922

6. Conclusions

In conclusion, in this work we present the airline crew pairing problem and how it can be formulated as a Set Covering Problem and as a Set Partitioning Problem. In addition, we describe the most common meta-heuristics that are used to solve the problem. Multiple genetic algorithms were implemented in order to solve the problem, which was formulated as an SCP and an SPP. The results of these implementations show that GAs are efficient at finding optimal or near-optimal solutions, especially for the SCP.

Acronyms and Abbreviations

SPP	Set Partitioning Problem
SCP	Set Covering Problem
EA	Evolutionary Algorithm
SI	Swarm Intelligence
ACO	Ant Colony Optimization
PSO	Particle Swarm Optimization
ABC	Artificial Bee Colony
MMAS	Min-Max Ant System
ACS	Ant Colony System
ANTS	Approximate Nondeterministic Tree Search
CP	Constraint Programming
AS	Ant System
TSP	Traveling Salesman Problem
MCS	Max Compatibility Selection
SOR	Subgroup Ordering Replacement
CUC	Cost-Based Uniform Crossover
OPT	Optimal
BKS	Best Known Solution
NF	Not Feasible

References

[1] I. Boussaïd et al., "A survey on optimization meta-heuristics, " Information Sciences 237 (2013) 82–117

[2] Broderick Crawford, Carlos Castro and Eric Monfroy, "A Constructive Hybrid Algorithm for Crew Pairing Optimization, " J. Euzenat and J. Domingue (Eds.): AIMSA 2006, LNAI 4183, pp. 45–55, 2006.

[3] H. Y. Quek, K. C. Tan and A. Tay, "A Discrete Particle Swarm Optimization Approach for Airline Crew Scheduling, " Department of Electrical and Computer Engineering, National University of Singapore 4, Engineering Drive 3, Singapore 117576.

[4] Guang-Feng Deng and Woo-Tsong Lin "Ant colony optimization-based algorithm for airline crew scheduling problem, " Expert Systems with Applications 38 (2011) 5787–5793.

[5] P.C. Chu and J.E. Beasley, "A Genetic Algorithm for the Set Partitioning Problem, " The Management School Imperial College London SW7 2AZ, England

[6] David Levine, "Application of a Hybrid Genetic Algorithm to Airline Crew Scheduling, " Computers Ops Res. Vol. 23, No. 6, pp. 547-558, 1996

[7] Harry Kornilakis and Panagiotis Stamatopoulos, "Crew Pairing Optimization with Genetic Algorithms, " Department of Informatics and Telecommunications University of Athens Panepistimiopolis, 15784 Athens, Greece

[8] Ketan Kotecha, Gopi Sanghani and Nilesh Gambhava, "Genetic Algorithm for Airline Crew Scheduling Problem Using Cost-Based Uniform Crossover, " S. Manandhar et al. (Eds.): AACC 2004, LNCS 3285, pp. 84–91, 2004

[9] Bahadır Zeren and İbrahim Özkol, "An Improved Genetic Algorithm for Crew Pairing Optimization, " Journal of Intelligent Learning Systems and Applications, 2012, 4, 70-80

[10] Ahmad E. Elhabashy, M. Hamdy Elwany, M. Nashat Fors, and Yasmine Abouelseoud, "Solving the Airline Crew Pairing Problem Using Genetic Algorithms, " CIE44 & IMSS'14 Proceedings, 14-16October 2014, Istanbul / Turkey, Pages: 2167-2181

[11] J.E Beasley, P.C. Chu, "A Genetic Algorithm for the set covering problem, " European Journal of Operational Research 94 (1996) 392-404

[12] Zhi-Gang Ren, Zu-Ren Feng, Liang-Jun Ke, Zhao-Jun Zhang, "New ideas for applying ant colony optimization to the set covering problem, " Computers & Industrial Engineering 58 (2010) 774–784

[13] Lucas Lessing, Irina Dumitrescu and Thomas Stützle, "A Comparison Between ACO Algorithms for the Set Covering Problem, " M. Dorigo et al. (Eds.): ANTS 2004, LNCS 3172, pp. 1–12, 2004.

[14] Sameh Al-Shihabi , Mazen Arafeh and Mahmoud Barghash, "An improved hybrid algorithm for the set covering problem ," Computers & Industrial Engineering 85 (2015) 328–334

[15] Shyam Sundar and Alok Singh, "A hybrid heuristic for the set covering problem, " Oper Res Int J (2012) 12:345–365