J Math Model Algor (2008) 7:25–42
DOI 10.1007/s10852-007-9071-8

Distance Preserving Recombination Operator for Earth Observation Satellites Operations Scheduling

Andrzej Jaszkiewicz

Received: 15 February 2005 / Accepted: 16 July 2007 /
Published online: 22 November 2007

Abstract The paper describes an adaptation of a memetic algorithm to the problem of scheduling operations of Earth observation satellites. The adaptation uses a systematic approach to the design of the recombination operator preserving important features common to both parents. The important features are identified experimentally on the basis of correlations between the value of the objective function and the similarity of good solutions. Our results indicate that this systematic approach reduces the effort needed to design a high quality recombination operator by avoiding not promising development directions.

Keywords Hybrid evolutionary algorithms · Recombination operators

Mathematics Subject Classifications (2000) 90C27

1 Introduction

This paper describes an adaptation of a memetic algorithm (MA) to the problem of scheduling operations of Earth observation satellites. The main element of this adaptation is a systematic approach to the design of the recombination operator on the basis of fitness–distance correlation tests. This approach helped to design a well-performing operator while avoiding intensive tests in a trial-and-error manner.

The problem considered here is that of ROADEF Challenge 2003 – an international competition organized by the French Society of Operations Research and Decision Analysis [19]. The problem has been formulated by two French space

A. Jaszkiewicz (✉)
Institute of Computing Science, Poznan University of Technology,
ul. Piotrowo 3a, 60-965 Poznan, Poland
e-mail: jaszkiewicz@cs.put.poznan.pl
URL: http://www-idss.cs.put.poznan.pl/~jaszkiewicz

agencies ONERA and CNES. It is based on real problems arising in the management of missions of Earth observation satellites. The problem has been simplified for the purpose of the challenge, still being a relatively complicated combinatorial problem.

The paper is organized in the following way. In the next section the adaptation of the MA algorithm to a given problem is outlined. In the third section, the challenge problem is described. The adaptation of the MA framework to the challenge problem is described in the fourth section. In the fifth section, the results of computational experiments are presented. The last section contains conclusions.

2 Systematic Approach to the Adaptation of the MA Algorithm to a Given Problem

Memetic algorithms (MAs) are relatively new metaheuristics that hybridize evolutionary algorithms with local search, or more generally, with other local search heuristics. Other frequently used names are Genetic Local Search or Hybrid Genetic Algorithms. Methods of this type often perform very well when applied to combinatorial optimization problems, e.g., the traveling salesperson problems (TSP) [17], the graph coloring problems [5], the quadratic assignment problems [18, 20], vehicle routing problem [9], and the set covering problem [1]. The methods proved to be successful in achieving a synergy between genetic/evolutionary methods and local optimization.

From one point of view the MA may be interpreted as an evolutionary algorithm working on a reduced search space containing local optima of a given problem only. If local optima can be generated effectively, this reduction of search space should improve the overall performance. Clearly, for many combinatorial problems local optima can be generated very efficiently. For example, it is well known that in the case of the TSP local search using two arcs exchange move converges in O(N) iterations (N being the number of nodes). This requires evaluation of O(N3) solutions. For the same problem, Boese et al. [2] estimated on the basis of their experiment that local optima are concentrated in a subregion being just $\frac{1}{10^{59}}$ of the total search space for instances with 100 nodes. For instances with 500 nodes this value was estimated to $\frac{1}{10^{468}}$. Thus, the use of local search can result in a significant reduction of the search space at a relatively low computational cost.

From another point of view, the MA may be interpreted as a version of multiple start (iterated) local search with improved starting solutions. Indeed, Jaszkiewicz and Kominek [10] noticed that in the MA adapted to a vehicle routing problem, both running time and the quality of solutions improves when local search is started from solutions produced by recombination algorithm. Furthermore, a recombination operator used in the MA does not need to produce very good solutions. It is enough if it generates solutions that are close in the search space to the good ones and can be efficiently improved by local search.

Although some metaheuristics, e.g. MA, are well recognized as effective tools for various optimization problems, the "no free lunch" (NFL) theorem [22] clearly states that there is no algorithm that outperforms all other algorithms when its performance is averaged over all possible objective functions. Thus, all optimization algorithms, including metaheuristics, are based on some, possibly implicit, assumptions; i.e. they are appropriate for some classes of problems. The NFL theorem is based on an

assumption of a uniform probability distribution over the set of all possible objective functions. The probability distribution corresponding to the "real-life" problems can be much different. The success of metaheuristics in many applications proves that their assumptions are often met in practice.

Metaheuristics define general schemes for the optimization procedures that have to be adapted for specific problems. One direct conclusion from the NFL theorem is that no such general scheme guarantees efficient optimization without appropriate adaptation. Thus, the way a given metaheuristic is adapted to a particular problem may have a crucial influence on its performance. This statement may be reinforced by an experiment conducted by Michalewicz for a set of evolutionary algorithms [15, 16]. The experiment showed that maximum efficiency of such an algorithm is obtained if it is totally adapted to a problem. Michalewicz concludes with words of Davis [3] that if we shrink from adding problem-specific knowledge to genetic algorithms it is very likely that they are worse than any reasonable optimization algorithm which takes such knowledge into account.

The process of adapting a metaheuristic to a given problem is, however, a challenging task even for problems with a simple definition. For example, efficient algorithms for TSP [17] are the result of many of years of research on this problem.

In our opinion, in order to reduce the time needed to adapt efficiently a metaheuristic to a given problem, some general adaptation patterns should be derived from the past successful works. This concept is similar to e.g. design patterns widely used in software engineering [6]. In fact, the schemes of metaheuristics may also be interpreted as design patterns.

In this paper, we use an adaptation pattern that allows systematic designing of a distance preserving recombination operator for a given problem. The distance preserving recombination operator [17] is an operator that guarantees (or at least makes an effort to guarantee) preservation in the offspring of important features common to both parents. At the same time, features considered to be not important, do not need (and in fact should not be) to be preserved even if they were common to both parents.

In order to explain this approach we will need to clearly define the terms "feature" and "feature realization." By *feature* we will understand a general attribute that may be used to describe solutions of a given problem (regardless of their representation in a given algorithm). For example, for the TSP the features could be arcs (pairs of consecutive nodes in the solution) or node positions in the sequence of nodes defining the solution. In the case of graph coloring problem the possible features may be colors of particular nodes, or pairs, or groups of nodes with the same color. One may use various features to describe solutions of a given problem.

By *feature realization* we will understand an instance of a given feature in a concrete solution. For example, in the TSP the realization of arcs feature may be an arc between nodes a and b, and the realization of positions feature may be node c being at the l-th position in the sequence of nodes defining the solution. Usually, a feature will have many realizations in a single solution. Now, we may state more precisely that a *distance preserving recombination* operator should preserve realizations of important features common to both parents.

The main question is what features are important for a given problem. Clearly, preservation of common realizations of all possible features for a given problem is not a good approach, since it would result in a very fast premature convergence

of the evolutionary algorithm. In order to identify the important features we use the concept of fitness–distance correlation test [10, 11]. This test checks if the quality of solutions is correlated to their similarity to other good solutions. Such correlation indicates that a trend may be noticed for better solutions to become more and more similar. Note, however, that fitness–distance correlation tests were originally proposed by as a measure of problem difficulty not as a mean of designing recombination operators [11].

Fitness–distance correlation has been noticed for a number of problems. Boese et al. [2] noticed it for the TSP. According to the results of Jones and Forrest [11], some numerical optimization problems also exhibit the property fitness–distance correlation. Merz and Freisleben [18] noticed fitness–distance correlation for the quadratic assignment problems. Jaszkiewicz and Kominek [10], and Kubiak [12] noticed fitness–distance correlation for two different variants of the vehicle routing problem.

Of course, similarity of solutions may be measured in many ways taking into account various features of solutions. We will assume that the similarity of two solutions is measured by the number (or percentage) of common realizations of a given feature. For the same problem the fitness–distance correlation may be noticed for some features but not for others. Furthermore, it may be stronger for some features than for others.

To test fitness–distance correlation we propose to calculate the correlation between the value of the objective function f for solution x and the average similarity of all solutions not worse than $f(\mathbf{x})$ denoted by $\hat{s}(f(\mathbf{x}))$. Assume that a set C of solutions is known. The average similarity is calculated as:

$$\hat{s}(f(\mathbf{x})) = \frac{\sum_{\mathbf{y},\mathbf{z}\in C | f(\mathbf{y})\geq f(\mathbf{x}) \wedge f(\mathbf{z})\geq f(\mathbf{x})} s(\mathbf{y},\mathbf{z})}{\left|\left\{(\mathbf{y},\mathbf{z})\right) \mid \mathbf{y},\mathbf{z}\in C \wedge f(\mathbf{y})\geq f(\mathbf{x}) \wedge f(\mathbf{z})\geq f(\mathbf{x})\right\}\right|}$$

where $s(\mathbf{y},\mathbf{z})$ is similarity of $\mathbf{y}$ and $\mathbf{z}$ and objective function f is assumed to be maximized. Since for the few best solutions in C, $\hat{s}(f(\mathbf{x}))$ would be calculated on the basis of a small number of solutions one could expect a high dispersion of this value in this case. Thus, we have decided not to calculate it for the best 20 solutions. The value of 20 is an arbitrary one, based on our observations that the dispersion of the results becomes noticeably lower for other solutions.

On the basis of previous works of Merz and Freisleben [17, 18], Jaszkiewicz and Kominek [10], and Kubiak [12] we formulate the following adaptation pattern for (hybrid) evolutionary algorithms:

1. Generate sets of good and diversified solutions for a set of instances of a given problem.
2. Formulate a number of hypotheses about solution features important for a given problem and define similarity measures based on these features.
3. For each feature and each instance test the importance of this feature by a correlation between the value of the objective function and the similarity of good solutions. The similarity is measured with respect to this feature.
4. Design a distance preserving recombination operator assuring (or aiming at) preservation of common instances of features for which positive correlations were observed. The operator may preserve common instances of several features.

To our knowledge this pattern was used for the first time by Merz and Freisleben [17] for the TSP. On the basis of results of the experiment of Boese et al. [2], Merz and Freisleben designed a very successful recombination operator preserving arcs common to both parents. Jaszkiewicz and Kominek [10] considered four features for a version of the vehicle routing problem and concluded that the recombination operator should preserve all of the features.

3 Scheduling of Earth Observation Satellites Operations

The subject problem is a simplification of a real problem in the management of Earth observation satellites. In particular, we just consider a single revolution of the satellite. Below, we define it formally.

Given:

- A number N_r of requests for satellite pictures that may be satisfied fully or partially (a request defines an area to be photographed),
- A number N_s of strips that can be used to acquire the requests,
- A number $N_{pa} = 2N_s$ of possible strips acquisitions (each strip can be acquired in by moving the camera in one of the two possible directions),

find a subset of acquisitions and their feasible schedule that maximizes the gain of the satellite operator.

Each request is defined by a number of strips. A strip is a relatively thin and long rectangle (approximately, of course, because of the Earth shape) that the satellite may acquire at a time moving the camera along it. The two possible acquisitions correspond to starting from one or another end of the strip. The acquisitions have durations and time windows in which they are physically possible. Furthermore, a switching time may be calculated for each pair of acquisitions. The gain corresponding to a given request is a nonlinear function of the percentage of the request realized (photographed). More precisely, it is a piece wise linear function going through points (0, 0), (0.4, 0.1), (0.7, 0.4), and (1, 1). This means, for example, that if 40% of the request surface is realized, only 10% of the gain corresponding to this request is achieved. The total gain is the sum of gains for particular requests. Some of the requests require stereoscopic pictures. In this case pairs of twin strips are defined. The twin strips have to be either realized both or not realized at all. A detailed description of the problem is publicly available at [19].

The challenge problem has some similarities with task scheduling on a single machine, with time windows and switching times [4]. However, in this case, not all acquisitions (tasks) have to be scheduled. In fact, for the considered instances only a small number of them are selected and scheduled. Furthermore, the objective function is different from those typically used in scheduling.

The problem resembles also the vehicle routing problem with time windows and service times [21]. In this case, acquisitions would correspond to nodes to be visited, their durations to service times at the nodes, and switching times to travel times between nodes. However, in most versions of the vehicle routing problem all nodes have to be visited.

An analogy to the knapsack problem may also be noticed [14]. In the challenge problem one needs to select a subset of items (acquisitions) maximizing their value

and subject to some constraints. However, both the constraints and the objective function are much more complicated than in knapsack problem.

Finally, there are some important similarities with an only recently studied trip planning problem [8] that also involves selection of attractions (acquisitions) and their scheduling. There are, however, significant differences in definition of some constraints and objective functions.

4 Adaptation of the MA to the Challenge Problem

Our adaptation of the MA to the challenge problem involves the definition of a method for finding initial solutions, the definition of a local search method, the definition of a recombination operator, and the selection of the general scheme of the MA algorithm. Below we define the elements in more detail.

4.1 Generation of Initial Solutions

The initial solutions are generated by a greedy insertion heuristic. In each iteration, the heuristic inserts a single acquisition to the solution. All acquisitions that can be inserted without violating constraints are evaluated. For each acquisition all feasible insertion positions in the current sequence of selected acquisitions are evaluated.

Insertion of an acquisition increases the gain objective but it also consumes the time of the satellite. The consumed satellite time (including switching time) may be different depending on the insertion position. Thus, the insertion of an acquisition at a given position is evaluated by the ratio of the gain increase over the satellite time change. In each iteration the greedy insertion heuristic selects the acquisition and insertion position with the highest evaluation.

The greedy insertion heuristic is a deterministic procedure (unless more than one insertion have the same best evaluation). In order to obtain a dispersed sample of initial solutions, the greedy insertion heuristic is not started from the empty solution. Instead a number of randomly selected acquisitions are inserted at randomly selected feasible positions before applying the insertion heuristic. The number of randomly selected acquisitions was set to $N_s/10$, however, the final method was very robust with respect to this parameter.

We identify feasible insertion positions by exploiting time constraints which resulted in a significant improvement of the efficiency of this method.

4.2 Local Search

Local search uses a neighborhood operator that inserts an unselected acquisition into the solution. If necessary, a number of already selected consecutive acquisitions are removed from the solution to maintain feasibility.

The new acquisition is inserted in place of removed ones. For each acquisition to be inserted, all potential minimal sets of consecutive acquisitions to be removed are considered. We attempt to reduce the set of acquisitions considered for removal taking into account time constraints.

A greedy version of the local search is used. The acquisitions to be inserted are analyzed starting from a random position and the first move that improves the gain is performed. Only feasible moves are performed.

4.3 Recombination Operators

We have identified five potential important features of solutions in the subject problem:

1. *Acquisitions selected for realization* (i.e. placed in the final solution). The hypothesis is that good solutions tend to have the same acquisitions selected. This hypothesis has been formulated on the basis of the analogy to the knapsack problem where selected items is known to be an important feature. The corresponding similarity measure counts the percentage of acquisitions being common to both parents.
2. *Requests selected for realization*. The hypothesis is that good solutions tend to have the same requests realized. Note that the requests may be realized only partially. Thus, two versions of the corresponding similarity measure were used, taking into account either the realized (photographed) surface in square kilometers, or the realized percentage of each request. This hypothesis has also been formulated on the basis of the analogy to the knapsack problem. Obviously, this feature is related to feature 1, however, since a given request may be realized by various acquisitions (partial realization, acquisitions in opposite directions) we have decided to also test this feature. The corresponding similarity measure counts the total surface or total normalized surface of common requests selected for realization. The total surface is expressed in surface units, while the normalized surface as a ration of the realized (photographed) surface over the total surface of a given request.
3. *Pairs of consecutive selected acquisitions*. The intuition behind this hypothesis is that switching times between some acquisitions are very small (the final position of an acquisition is very close to the beginning position of the other one) and they should be usually scheduled consecutively in good solutions. This is an analogy to the vehicle routing problem. The corresponding similarity measure counts the percentage of pairs of acquisitions selected in both parents and being consecutive in both solutions.
4. *Sequences of (consecutive or not) pairs of selected acquisitions*. The hypothesis is that good solutions tend to order selected acquisitions similarly. The corresponding similarity measure counts the percentage of pairs of acquisitions selected in both parents and being sequenced in the same way in both solutions.
5. *Starting time windows of acquisitions resulting from the solution*. The solution is defined as a schedule of selected acquisitions. The schedule, in general, does not define precise starting times for each acquisition but rather some time windows for the starting times. These time windows are, of course, included in the original time windows defined in the problem instance. The hypothesis is that these time windows tend to become similar in good solutions. The corresponding similarity measure is the average distance between central points

of time windows corresponding to the same selected acquisitions. There are two versions of this measure taking into account the distances in seconds or distances normalized with respect to the time window duration.

Fitness–distance correlation tests have been performed on six instances delivered by the challenge organizers and based on real data. Some basics characteristics of the instances are summarized in Table 1.

For each instance, 1,000 local optima were generated. Equal solutions were then filtered out. Since, it is expected that the populations of the evolutionary algorithm will contain mainly very good solutions the calculations have also been made for the 200 best solutions only. The correlations are presented in Table 2 and Table 3. Each row corresponds to an instance and each column to a similarity measure. Each entry is a correlation between $f(\mathbf{x})$ and $\widehat{s}(f(\mathbf{x}))$. The results for a single (largest) instance are also presented in Fig. 1. Each point corresponds to a single solution. The horizontal axis represents the objective function value of a given solution $f(\mathbf{x})$, and the vertical axis represents $\widehat{s}(f(\mathbf{x}))$. The figure includes all solutions including the 20 best in C. One may notice a high dispersion of these values for the best few solutions.

In most cases significant correlations have been noticed. In general, the correlations are stronger when only the first 200 solutions are taken into account. On the basis of the results it has been concluded that the recombination operator should preserve the following features:

- Acquisitions common to both parents (note that this should also assure preservation of requests realization, thus there is no need to consider this feature separately),
- Sequences of any two pairs of acquisitions common to both parents,
- Pairs of consecutive selected acquisitions common to both parents,
- Time windows positions.

However, in order to test the systematic approach to the design of recombination operators, several recombination operators preserving various features have been designed. The operators are described below. Note, however, that only one of the operators has been designed for the purpose of the challenge (operator C). The other operators were designed afterwards for the purpose of this paper.

Table 1 Characteristics of the instances used in fitness–distance correlation tests

Instance code	Number of requests	Number of strips
2_13_111	68	106
4_17_186	77	147
3_25_22	150	342
2_15_170	218	295
2_26_96	336	483
2_27_22	375	534

Table 2 Results of the fitness distance correlation tests for all 1,000 solutions

Key						
Instance	2_13_111	4_17_186	3_25_22	2_15_170	2_26_96	2_27_22
SelAcq	0.972	0.929	0.933	0.927	0.902	0.885
ReqRea-ns	0.952	−0.119	0.848	0.945	0.920	0.933
ReqRea-s	0.978	−0.276	0.959	0.946	0.887	0.845
ConAcq	0.936	0.855	0.918	0.937	0.866	0.603
SeqAcq	0.981	0.775	0.907	0.978	0.874	0.942
Twin-nd	0.966	0.212	0.853	0.930	0.824	0.561
TWin-d	0.965	0.922	0.875	0.938	0.781	0.713

Instance instance code, *SelAcq* selected acquisitions, *ReqRea-ns* requests selected for realization (normalized surface), *ReqRea-s* requests selected for realization (surface), *ConAcq* pairs of consecutive selected acquisitions, *SeqAcq* sequences of pairs of selected acquisitions, *TWin-nd* starting time windows (normalized distances), *TWin-d* starting time windows (distances).

Operator A

This operator preserves acquisitions common to both parents. Its steps are summarized below:

1. Find the set of acquisitions selected in both parents.
2. Consider the acquisitions in a random order and insert each of them at the best possible position.
3. Add further acquisitions using the greedy insertion heuristic.

Operator B

This operator preserves acquisitions common to both parents and sequences of any two pairs of acquisitions common to both parents. Its steps are summarized below:

1. Find a set of acquisitions selected in both parents.
2. Schedule these acquisitions preserving common sequences.
3. Add further acquisitions using the greedy insertion heuristic.

In step 2 of this operator, the algorithm presented in Pseudocode 1 is used. The algorithm builds a new sequence S from two sequences $S1$ and $S2$. If element a

Table 3 Results of the fitness distance correlation tests for the best 200 solutions

Key						
Instance	2_13_111	4_17_186	3_25_22	2_15_170	2_26_96	2_27_22
SelAcq	0.957	0.950	0.958	0.959	0.985	0.988
ReqRea-ns	0.823	0.857	0.964	0.305	0.975	0.978
ReqRea-s	0.885	0.630	0.621	0.972	0.980	0.973
ConAcq	0.944	0.929	0.932	0.887	0.985	0.969
SeqAcq	0.872	0.882	0.513	0.814	0.954	0.506
Twin-nd	0.975	0.874	0.936	0.972	0.971	0.712
TWin-d	0.970	0.941	0.984	0.963	0.970	0.783

Instance instance code, *SelAcq* selected acquisitions, *ReqRea-ns* requests selected for realization (normalized surface), *ReqRea-s* requests selected for realization (surface), *ConAcq* pairs of consecutive selected acquisitions, *SeqAcq* sequences of pairs of selected acquisitions, *TWin-nd* starting time windows (normalized distances), *TWin-d* starting time windows (distances).

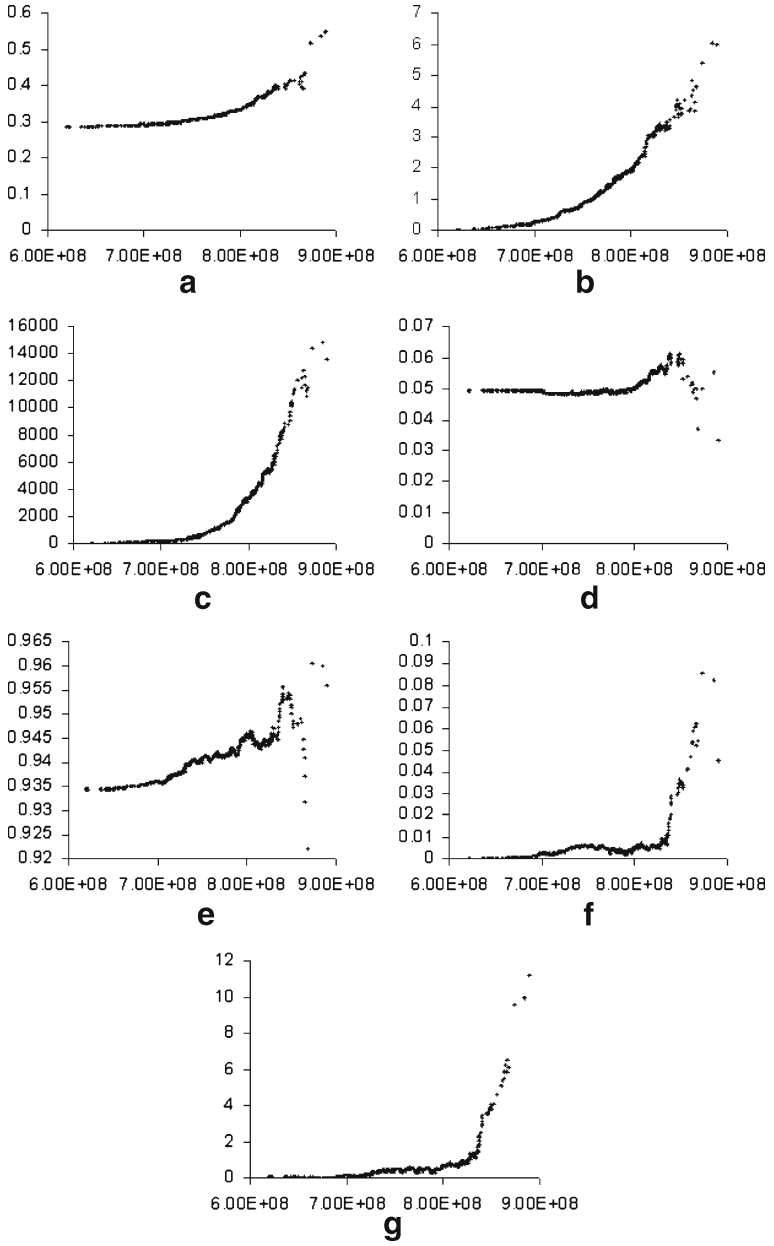

Fig. 1 Results for instance 2_27_22. The *horizontal axis* represents the objective function value of a given solution $f(\mathbf{x})$, and the *vertical axis* represents $\widehat{s}(f(\mathbf{x}))$. Key: **a** selected acquisitions, **b** requests selected for realization (normalized surface), **c** requests selected for realization (surface), **d** pairs of consecutive selected acquisitions, **e** starting time windows (normalized distances), **f** starting time windows (distances)

precedes element b in both sequences then a precedes b also in the new sequence, otherwise the sequence of the two elements will depend on the greedy insertion heuristic.

```
let S1 and S2 be sequences of common acquisitions in parents 1 and 2, respectively
let S be an empty sequence
repeat
    draw at random either S1 or S2
    if the acquisition at the first position in the selected sequence is not yet scheduled
        schedule at the end of S the acquisition at the first position in the selected sequence
    remove the acquisition at the first position from the selected sequence
until S1 and S2 are empty
```

Pseudocode 1 Sequence recombination

Operator C

This operator preserves acquisitions common to both parents, sequences of any two pairs of acquisitions if the same in both parents, and pairs of consecutive selected acquisitions if the same in both parents. Its steps are summarized below:

1. Find a set of acquisitions selected in both parents.
2. Schedule these acquisitions preserving common sequences.
3. Mark as locked those consecutive pairs of acquisitions that were also consecutive in both parents.
4. Add further acquisitions using the greedy insertion heuristic with the additional condition that locked pairs of consecutive acquisitions cannot be broken.

Operator D

This operator preserves only acquisitions common to both parents and time windows positions. Its steps are summarized below:

1. Find a set of acquisitions selected in both parents.
2. Reduce the original time windows of common acquisitions.
3. Consider the acquisitions in a random order and insert each of them at the best possible position.
4. Add further acquisitions using the greedy insertion heuristic.

In step 2 of this operator, the feasible time window for each acquisition common to both parents is reduced with respect to the original time windows defined in the problem instance. The beginning of the time window t_0' is set to:

$$t_0' = \frac{t_0 + \min\left(t_0^1, t_0^2\right)}{2}$$

where t_0 is the original beginning of the time window, t_0^1 and t_0^2 are beginnings of the time windows corresponding to this acquisition in parents 1 and 2, respectively.

The end of the time widow t'_e is set to:

$$t'_e = \frac{t_e + \max\left(t_e^1, t_e^2\right)}{2}$$

where t_e is the original end of the time window, t_e^1 and t_e^2 are ends of the time windows corresponding to this acquisition in parents 1 and 2, respectively.

Operator E

This operator preserves acquisitions common to both parents, sequences of any two pairs of acquisitions if the same in both parents, pairs of consecutive selected acquisitions if the same in both parents, and time windows positions. Its steps are summarized below:

1. Find a set of acquisitions selected in both parents.
2. Reduce the original time windows of common acquisitions.
3. Schedule these acquisitions preserving common sequences.
4. Mark as locked those consecutive pairs of acquisitions that were also consecutive in both parents.
5. Add further acquisitions using the greedy insertion heuristic with the additional condition that locked pairs of consecutive acquisitions cannot be broken.

We formulate a hypothesis that operator E should assure the best performance of the MA, since it uses all important features of the solutions. Note, however, that in the challenge, operator C was used. This was caused by the fact that a systematic error has been made by the author in calculation of the correlations for time windows related features. In result, the positive correlations were not noticed during the preparation of algorithm for the challenge.

4.4 Local Search Improvement

The performance of local search has been further improved with the concept of candidate moves. The idea is to heuristically identify a set of promising candidate moves. Non-candidate moves may also be evaluated, but the evaluation probability is 10% only.

The definition of candidate moves is strictly related to the recombination operators and the importance of the "acquisitions selected for realization" feature. Candidate moves are the moves that insert acquisitions that were present in at least one of the parents. The use of candidate moves significantly improved efficiency of local search without any noticeable deterioration of the results.

4.5 General Algorithm

The above described operators have been used within a general scheme of the algorithm which could be classified as a hybrid evolutionary algorithm with full elitism and island population model. The islands create a directed ring and periodically exchange their best solutions (see Fig. 2). The details of the algorithm are presented in Pseudocode 2. The island model has been selected in order to reduce dispersion of results being achieved by a corresponding algorithm with a single population.

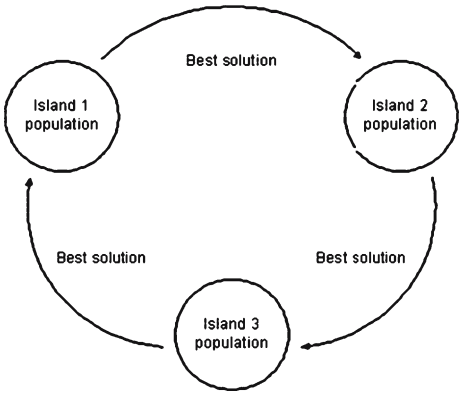

Fig. 2 Island population model

The dispersion or results in the latter case was very high. The introduction of the island model resulted in a significant improvement of the results variation, although the best results remained at the same level. Of course, it may be the case that other approaches like reduction of the elitism would also be helpful but they were not tested.

Parameters: Number of islands, Population size for each island - P, Maximum running time - *MRT*, Migration period [number of iterations] - *MP*, Local search probability - *LSP*

Generation of initial populations

for each island

  repeat P times

    generate new solution **x** with the insertion heuristic

    improve **x** with local search

Main loop

while running time does not exceed *MRT*

  for each island

    draw two different solutions $\mathbf{x}_1$ and $\mathbf{x}_2$ (parents) from the island population with uniform probability

    recombine $\mathbf{x}_1$ and $\mathbf{x}_2$ obtaining **x**

    with *LSP* probability improve **x** with local search

    if **x** is better than the worst solution in the island population and different from all solutions in island population

      add **x** to the island population and remove the worst solution from it

  if the number of iterations is equal to *MP*

    for each island

      select the best solution from the island population **x**

      if **x** is better than the worst solution in the neighbor island population and different from all solutions in neighbor island population

add **x** to the neighbor island population and remove the worst solution from it

Pseudocode 2 General scheme of the MA with island population model

The parameters were set experimentally on the basis of the best choice principle. The number of islands was set to 5, population size for each island to 50, the maximum running time to 300s (PC~1000MHz), the migration period to 100 iterations, and local search probability to 2,5%.

5 Results

5.1 Challenge Results

There were two stages of the challenge – qualification stage and final stage. All registered teams were allowed to take part in the qualification stage. Among thirty one registered teams eighteen decided to submit their programs for the qualification stage. Eleven teams were qualified to the final stage. In the qualification stage, eight instances known in advance to the participants were used. Ten instances used in the final stage were not known to the participants. In both stages, the challenge organizers performed their own calculations using programs delivered by the participants. The organizers specified the maximum running time. The results were compared against some reference results obtained for the same instances by an existing algorithm (its details are not known to the participants). Note, that all participants of the final stage obtained results much better than the reference results.

Both deterministic and nondeterministic methods were allowed. The nondeterministic methods were run ten times on each instance. The overall evaluation of deterministic methods was calculated in the following way:

$$\text{Evaluation} = 100 \times \frac{1}{10} \sum_{i=1}^{10} \frac{(\text{ValSol}_i - \text{Val}_i)}{\text{Val}_i}$$

where ValSol_i is the objective function value generated by the algorithm for the i-th instance, Val_i is the reference objective value for this instance.

The overall evaluation of nondeterministic methods was calculated in the following way:

$$\text{Evaluation} = 100 \times \frac{1}{100} \sum_{i=1}^{10} \sum_{j=1}^{10} \frac{(\text{ValSol}_{ij} - \text{Val}_i)}{\text{Val}_i}$$

where ValSol_{ij} is the objective function value generated by the algorithm for the i-th instance in j-th run.

The summary of the final stage results is presented in Table 4. Note that all of the results improve the reference results by at least 23%. Note also, that all evaluations except of the last team are relatively close. The gap for the ten best teams is lower than 3% which is relatively low value in comparison to the improvement over the reference results.

Table 4 Results of the final stage

Summary											
Team	1 (best)	2	3	4	5 (author)	6	7	8	9	10	11
Evaluation	31.76	30.84	30.28	30.1	29.91	29.88	29.84	29.69	29.22	29.08	23.56

5.2 Additional Evaluation of the Recombination Operators

In order to further evaluate the systematic approach to the design of the recombination operators, the five operators described in Section 4.3 have been tested in an additional experiment. The experiment has been performed on the set of instances used in the final stage of the challenge. The same evaluation procedure has been used. Ten runs of each algorithm have been made. The running time was the same (300 s) on a comparable machine, slightly slower than 1,000 MHz in fact.

In addition, the MA algorithm was compared with iterated generation of initial solutions (IGIS) with the same time limit. The initial solutions were generated by the greedy insertion heuristic and then local search was applied with 2.5% probability, i.e. the same as used after the recombination. The results are summarized in Table 5. In addition, we used t-test in order to test if the differences in performance are

Table 5 Results of the additional experiment (standard deviations are given in brackets)

Instance	Ref	OpA	OpB	OpC	OpD	OpE	IGIS
2_28_111	641971585	28.81	35.15	34.81	32.79	35.32	22.93
		(1.635)	(0.655)	(0.782)	(1.683)	(0.529)	(1.729)
2_28_140	688993450	19.82	20.24	20.30	20.40	20.71	18.15
		(0.384)	(0.082)	(0.238)	(0.295)	(0.303)	(0.270)
2_28_155	754868831	20.74	24.32	24.26	24.26	24.34	15.25
		(0.463)	(0.191)	(0.000)	(0.000)	(0.233)	(1.478)
2_28_170	643688604	39.97	41.20	41.93	38.58	42.26	34.49
		(0.699)	(2.013)	(1.855)	(0.960)	(1.645)	(1.654)
2_28_170	643688604	39.97	41.20	41.93	38.58	42.26	34.49
		(0.699)	(2.013)	(1.855)	(0.960)	(1.645)	(1.654)
2_28_37	632923003	47.72	50.78	51.19	47.39	50.88	36.37
		(1.206)	(0.915)	(1.099)	(1.232)	(1.390)	(1.829)
2_28_66	686621650	32.37	35.87	35.55	33.24	36.22	26.42
		(2.232)	(0.976)	(0.857)	(2.593)	(1.088)	(2.888)
2_28_7	714878886	28.02	33.43	34.45	33.84	34.58	22.66
		(4.323)	(2.492)	(2.015)	(2.570)	(2.302)	(3.890)
2_28_81	728186221	15.85	20.54	20.25	19.59	20.69	11.20
		(1.352)	(0.485)	(0.380)	(0.377)	(0.000)	(1.060)
3_28_155	380505982	16.12	21.43	21.44	21.13	21.44	15.37
		(2.379)	(0.021)	(0.000)	(0.958)	(0.000)	(2.145)
3_28_96	396698443	14.94	15.48	15.48	14.99	15.48	13.97
		(0.212)	(0.000)	(0.000)	(0.000)	(0.000)	(1.170)
Average		26.44	29.84	29.96	28.62	30.19	21.68
		(1.488)	(0.783)	(0.723)	(1.067)	(0.749)	(1.811)

Instance instance code, *Ref* reference value, *OpA* MA operator A, *OpB* MA operator B, *OpC* MA operator C, *OpD* MA operator D, *OpE* MA operator E, *IGIS* integrated generation of initial solutions

statistically significant. For each instance and each algorithm series of 10 results have been compared. The differences were statistically significant at level 0.05 in the case of all of the ten instances for the following pairs of algorithms: MA operator B > IGIS, MA operator C > IGIS, MA operator E > IGIS, MA operator D > IGIS, MA operator C > MA operator A, MA operator E > MA operator A. In 9 cases the following differences were significant: MA operator A > IGIS, MA operator B > MA operator A.

On the basis of the results, the following observation could be made:

- All versions of the MA perform better than iterated generation of initial solutions.
- Operators preserving more significant features perform in general better than operators preserving less features.
- MA with operator E preserving all features is, on average, the best performer, however, on many instances the differences between MA with operators B, C and D are relatively low and not statistically significant.

After finishing the main experiment some additional experiments have been made with a goal of finding a similarity measure with no noticeable correlation with the objective function value. However, such correlations were found for any realistic measures (i.e. measures that do not transform solution features to some random values), e.g. the percentage of common acquisition positions or the percentage of common bits in time windows starting/ending times. This shows that good solutions of this problem converge in many aspects. Note, however, that in the case of the additional measures, dispersions of their values were very low. Of course, we do not claim that such measures do not exist. There still may exist some other realistic measures that were not tested by us, with no correlation with the objective function value.

6 Conclusions

The main goal of the systematic approach to the adaptation of the MA algorithm to a given problem presented in this paper is to reduce the effort needed to design a high quality algorithm. By effort we mean both designer time and the time of the computational experiments. Of course, one may argue that the proposed systematic approach is also a trial-and-error method alike trial-and-error evaluation of various recombination operators. However, the number of potential features to be tested for a given problem is much lower than the number of possible recombination operators. Furthermore, tests for a given feature require much shorter calculations than evaluations of recombination operators with the full algorithm.

In the described case, the systematic approach allowed avoiding many not promising paths in the development of the algorithm. The average results of the designed algorithm are less than 2% below the best results achieved in the challenge.

The results of the additional experiments indicate that preservation of significant common features indeed improves performance of the recombination operators, however, with inclusion of additional features the improvements in the performance become lower.

The algorithm described did not produce best results in the ROADEF Challenge. The author is at the opinion, however, that there is a place in the literature for papers studying influence of particular design elements on the performance of optimization. This paper focuses on the design of recombination operators. On the other hand, papers presenting the best results often propose algorithms that differ from others in many aspects, and it is difficult to judge which of the elements are crucial for the overall performance. The results of the proposed MA algorithm could probably be further improved by investing more effort for example in the design of local search. In fact, the winning algorithm in ROADEF Challenge 2003 [13] used an advanced local search with a large long term memory within the general scheme of simulated annealing.

In the future we would like to use fitness–distance correlation tests in systematic design of other promising methods based on the idea of preservation of some common features of good solutions, e.g. path relinking [7] or probe method.

There is also a need for more systematic and based on stronger theoretical basis approach for quantitative evaluation of similarity measures. The approach applied in this paper is rather an "engineering one" based on both correlations and visual evaluation. The quantitative approach should probably take also into account the range of changes of a given measure.

Acknowledgement This work was supported by State Committee for Scientific Research, KBN research grant no. 3 T11F 004 26.

References

1. Beasley, J.E., Chu, P.C.: A genetic algorithm for the set covering problem. Eur. J. Oper. Res. **94**, 392–404 (1996)
2. Boese, K., Kahng, A., Muddu, S.: A new adaptive multistart technique for combinatorial global optimization. Oper. Res. Lett. **16**, 101–113 (1994)
3. Davis, L.: Adapting operator probabilities in genetic algorithms. In: Schaffer (ed.) Proceedings of the Third International Conference on Genetic Algorithms and Their Applications Morgan Kaufmann, pp. 61–69. San Mateo, CA (1989)
4. Desrosiers, J., Dumas, Y., Solomon, M.M., Soumis, F.: Time constrained routing and scheduling. In: Ball, M.O., Magnanti, T.L., Monma, C.L., Nemhauser, G.L. (eds.) Network Routing Vol 8 of Handbooks in Operations Research and Management Science, pp. 35-139. North-Holland, Amsterdam (1995)
5. Galinier, P., Hao, J.-K.: Hybrid evolutionary algorithms for graph coloring. J. Comb. Optim. **3**(4), 379–397 (1999)
6. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns Elements of Reusable Object-Oriented Software. Addison-Wesley (1994)
7. Glover, F.: Scatter search and path relinking. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimization, pp. 297–316. McGraw-Hill, London (1999)
8. Godart, J.M.: Combinatorial optimisation based decision/support system for trip planning. In: Proceedings of the 1999 International Conference on Information and Communication Technologies in Tourism Innsbrück (1999)
9. Homberger, J., Gehring, H.: Two evolutionary meta-heuristics for the vehicle routing problem with time windows. INFOR **37**, 297–318 (1999)
10. Jaszkiewicz, A., Kominek, P.: Genetic local search with distance preserving recombination operator for a vehicle routing problem. Eur. J. Oper. Res. **151**(2), 352–364 (2003)
11. Jones, T., Forrest, S.: Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In: Eshelman, L. (ed.) Proceedings of the Sixth International Conference on Genetic Algorithms, pp. 184–192. Morgan Kaufmann, San Francisco, CA (1995)

12. Kubiak, M.: Systematic construction of recombination operators for the vehicle routing problem. Foundation of Computing and Decision Sciences **29**, 3 (2004)
13. Kuipers, E.: An algorithm for selecting and timetabling requests for an earth observation satellite. Bulletin de la Socieete Française de Recherche Operationnelle et d'Aide a la Decision **11**, 7–10 (2003)
14. Martello, S., Toth, P.: Knapsack Problems, Algorithms and Computer Implementations. Wiley, England (1990)
15. Michalewicz, Z.: A hierarchy of evolution programs: an experimental Study. Evol. Comput. **1**(1), 51–76 (1993)
16. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer, Berlin Heidelberg (1992)
17. Merz, P., Freisleben, B.: Search for the TSP: new results. In: Proceedings of the 1997 IEEE International Conference on Evolutionary Computation IEEE Press, pp. 159–164 (1997)
18. Merz, P., Freisleben, B.: Fitness landscape analysis and memetic algorithms for the quadratic assignment problem. IEEE Trans. Evol. Comput. **4**(4), 337–352 (2000)
19. ROADEF: http://www.prism.uvsq.fr/~vdc/ROADEF/CHALLENGES/2003/. The ROADEF Challenge (2003)
20. Taillard, E.D.: Comparison of iterative searches for the quadratic assignment problem. Location Sci. **3**, 87–105 (1995)
21. Toth, P., Vigo, D.: The vehicle routing problem. SIAM Monographs on Discrete Mathematics and Applications (2002)
22. Wolpert, D.H., Macready, W. G.: No free lunch theorem for optimization. IEEE Trans. Evol. Comput. **1**(1), 67–82 (1997)