A decomposition approach for the Frequency Assignment Problem

Gualtiero Colombo

School of Computer Science, Cardiff University, Cardiff,

g.colombo@cs.cf.ac.uk

PhD Thesis, April 2008

Summary

The Frequency Assignment Problem (FAP) is an important optimization problem that arises in operational cellular wireless networks. Solution techniques based on meta-heuristic algorithms have been shown to be successful for some test problems but they have not been usually demonstrated on large scale problems that occur in practice.

This thesis applies a problem decomposition approach in order to solve FAP instances with standard meta-heuristics. Three different formulations of the problem are considered in order of difficulty: Minimum Span (MS-FAP), Fixed Spectrum (MS-FAP), and Minimum Interference FAP (MI-FAP). We propose a decomposed assignment technique which aims to divide the initial problem into a number of subproblems and then solves them either independently or in sequence respecting the constraints between them. Finally, partial subproblem solutions are recomposed into a solution of the original problem.

Standard implementations of meta-heuristics may require considerable runtimes to produce good quality results whenever a problem is very large or complex. Our results, obtained by applying the decomposed approach to a Simulated Annealing and a Genetic Algorithm with two different assignment representations (direct and order-based), show that the decomposed assignment approach proposed can improve their outcomes, both in terms of solution quality and runtime. A number of partitioning methods are presented and compared for each FAP, such as clique detection; partitioning based on sequential orderings; and novel applications of existing graph partitioning and clustering methods adapted for this problem.

Contents

1 Introduction **1**

1.1 The Frequency Assignment Problem (FAP) 2

1.2 Problem formulation 2

1.3 Types and models of FAP 4

1.3.1 Minimum Span-FAP 4

1.3.2 Fixed Spectrum-FAP 5

1.3.3 Minimum Interference-FAP 6

1.4 Other formulations and models 7

1.5 Solving large FAP instances 8

2 Literature review **11**

2.1 Exact methods and lower bounds 12

2.2 Meta-Heuristics for FAP 15

2.2.1 Constructive methods 16

2.2.2 Improvement methods 17

2.2.3 Evolutionary algorithms 19

2.2.4 Shortcomings of other existing approaches 22

2.2.5 Summary of meta-heuristics approaches 24

2.3 Problem decomposition for FAP 24

2.3.1 Decomposition combined with exact methods 25

2.3.2 Decomposition combined with heuristic methods 26

2.4 Benchmarks for FAP 28

3 Decomposition and assignment algorithms 31

3.1 Decomposed assignment approach 32

3.2 Decomposition algorithms 36

3.2.1 Random 37

3.2.2 Geographical 38

3.2.3 Minimum-cut 39

3.2.4 Cliques 41

3.2.5 Generalized degree 47

3.2.6 Graph clustering 50

3.2.7 Graph partitioning 52

4 Evaluation and benchmarks 62

4.1 Evaluation of a decomposition 63

4.1.1 Quality 63

4.1.2 Computational complexity 64

4.1.3 Trade-off between quality and runtime 66

4.2 Benchmark data sets 71

4.2.1 Benchmarks for MS-FAP/FS-FAP 71

4.2.2 Benchmarks for MI-FAP 77

5 Minimum Span FAP 88

5.1 Heuristic algorithms 89

5.2 Decomposition algorithms 93

5.3 Experimental results 95

6 Fixed Spectrum FAP 103

6.1 Heuristic algorithms 104

6.2 Decomposition algorithms 107

6.3 Experimental results 108

7 Minimum Interference-FAP **118**

7.1 Heuristic algorithms 119

7.1.1 Simulated Annealing 120

7.1.2 Genetic Algorithm 121

7.1.3 Multi objective approach 125

7.2 Decomposition algorithms 129

7.2.1 Graph partitioning 129

7.2.2 Graph clustering 131

7.2.3 Other decompositions 132

7.3 Experimental results 135

7.3.1 Comparison of decomposition algorithms 136

7.3.2 GPFAP results 140

7.4 Markov clustering 166

7.5 Trade-off between quality and runtime 170

7.5.1 Distribution of interference 170

7.5.2 Runs with variable number of loops 172

7.5.3 Cost-runtime trade-off 181

8 Conclusion and future work **183**

8.1 Conclusion 184

8.2 Future work 188

A Extended results **189**

A.1 Extended results for MS-FAP 190

A.2 Extended results for FS-FAP 193

A.3 Extended results for MI-FAP 196

B Condor Pool **204**

B.1 The Cardiff University Condor Pool 205

C Tuning of the meta-heuristics **207**

C.1 Tuning for FS-FAP 208

C.2 Tuning for MI-FAP 216

C.2.1 SA 216

C.2.2 GA 219

Chapter 1

Introduction

1.1 The Frequency Assignment Problem (FAP)

In a wireless network transmitters and receivers communicate via signals encoded on specific frequency channels. When adjacent transmitters use similar channels they may cause unacceptable interference. Thus a channel separation based on some interference measure is required for transmitters which are geographically close. The *Frequency Assignment Problem* (FAP) is an optimization problem which aims to assign frequencies to transmitters in as efficient way as possible, either in terms of interference or the amount of spectrum used.

In this thesis we will focus mainly on GSM networks which represent the second generation of digital cellular radio systems. However, the results and techniques proposed are more generally applicable. Although there are many forms of the FAP, roughly speaking the planning of a radio network consists of assigning the base stations a signal which is powerful enough to guarantee adequate communication, without causing severe interference between transmitters. These two requirements are in strong conflict in the particular case of GSM networks. Consequently, depending on the level of interference which can be considered acceptable, a required frequency separation can be specified for each pairs of transmitters. Alternatively, pairs of transmitters can be assigned numerical values which represent the acceptable interference that arises between them. In this case, the sum of the interference produced among all pairs of transmitters in the network should be minimized in the final frequency assignment.

1.2 Problem formulation

The FAP in its basic formulation considers only channel separations as constraints. Furthermore, it uses a *binary constraint* model as a measure of interference in which constraints are expressed between pairs of transmitters and specify the minimum separation of frequency channels that guarantees acceptable interference.

Formally, the FAP can be modelled by a unordered weighted graph $G(V, E)$, called the interference graph, which consists of a finite set of vertices V, representing transmitters, and a finite set of edges

$$E \subseteq \{ uv \mid u, v \in V \}$$

joining unordered distinct pairs of vertices. Each edge uv has an associated weight $c_{uv} \in \{0, 1, 2, \dots\}$ which is an integer value giving the channel separation required for the transmitters represented by its end points u, v.

Definition 1.1 *Given an allocation of allowed channels $F = \{1, 2, \dots, k\}$ and a frequency assignment $f : V \rightarrow F$ we define as* interfering transmitters *in the assignment F each pair of vertices $u, v \in V$ for which*

$$|f(v) - f(u)| < c_{uv}.$$

Definition 1.2 *An assignment is defined as a* zero-violation assignment *f if*

$$|f(v) - f(u)| \geq c_{uv} \quad \forall uv \in E$$

For some problems, for every vertex $v \in V$ we can specify a set of *blocked channels*

$$B_v \subseteq F$$

that cannot be assigned to the transmitter. Hence, we can represent the entire network by the 5-tuple $N = (V, E, F, \{B_v\}_{v \in V}, \{c_{uv}\}_{uv \in E})$.

1.3 Types and models of FAP

The FAP has been presented in the literature with many different formulations. However, two types of FAP are most commonly considered:

- the *Minimum Span FAP* (MS-FAP), which aims to minimize the range of frequencies used while respecting all of the constraint violations.
- the *Fixed Spectrum FAP* (FS-FAP) in which the domain of frequencies is instead limited to a fixed range. As a consequence it may not be possible to find a zero-violation assignment. The problem is formulated as minimizing a cost function which represents a measure of the global interference among all the network.

FS-FAP also includes a further refinement known as the *Minimum Interference FAP* (MI-FAP) in which the constraints are divided into the two categories of *hard constraints*, which must be respected in the final channel assignment, and *soft constraints* defined in terms of penalties. The latter group represents the amount of acceptable (yet still undesirable) interference between pairs of transmitters and their global sum across all the network constitutes the object to be minimized. In the following we will give the mathematical models which have been adopted in this thesis to solve the three categories of FAP outlined above.

1.3.1 Minimum Span-FAP

The *span* of an assignment is defined as the difference between the largest and the smallest channel used. In the MS-FAP the domain of frequencies is not bounded, i.e $F = \mathbb{Z}$ and $B_v = \emptyset \quad \forall v \in V$ and we are searching among zero-violation assignments. The problem is defined as following:

Problem 1.1 *Given an allocation of frequencies* $F = \{1, 2, \ldots, K\}$ *the* Minimum Span Frequency Assignment Problem (MS-FAP) *aims to determine a* zero-violation assignment $f : V \rightarrow F$ *which minimizes:*

•

$$O_{MS}(f) = \max_{v} f(v) - \min_{u} f(u) \qquad v,\ u \in V$$

A variant of the MS-FAP is the *Minimum Order FAP (MO-FAP)* in which the object of minimization is the number of distinct frequencies used in the network.

1.3.2 Fixed Spectrum-FAP

In the FS-FAP the spectrum of available channels is restricted to a limited range. Each of the constraints is a *hard constraint* which defines the channel separation that must be respected in the optimal solution in order to avoid interference. However, it may not be possible to satisfy all the constraints and some interference becomes therefore unavoidable.

The problem is formalised as finding an assignment which minimizes a cost function representing the global interference among all the network. The cost function is defined either as the number of interfering transmitters or as a measure of the size of violations among all the interfering transmitters. The latter is the objective that we have adopted in this thesis. Formally:

Problem 1.2 *Given a pair of transmitters u, v we define the cost of the associated constraint as:*

$$\varphi_{FS}(f, uv) = \begin{cases} c_{uv} - |f(u) - f(v)| & if\ |f(u) - f(v)| < c_{uv} \\ \\ 0 & otherwise \end{cases}$$

Hence given an allocation of frequencies $F = \{1, 2, \ldots, K\}$ the Fixed Spectrum Frequency Assignment Problem (MI-FAP) *aims to produce an assignment $f : V \rightarrow F$ which respects the blocked channel constraints, that is $f(v) \in F \backslash B_v \ \ \forall v \in V$,* *and minimizes:*

$$O_{FS}(f) = \sum_{uv\in E} \varphi_{FS}(f, uv)$$

1.3.3 Minimum Interference-FAP

In the MI-FAP, besides *hard constraints* c_e which represent the required separation between channels in order to avoid the interference which must be respected by any feasible solution, another category of weights known as *soft constraints* is associated with every edge $uv \in E$. These weights are expressed in terms of penalties which represent the probabilistic acceptable interference between pairs of transmitter which transmit on the same channel, (c_{uv}^{coch}) or on adjacent channels (c_{uv}^{adj}). Note that these values can be zero. As a consequence, the network is now represented by the 6-tuple $N = (G, F, \{B_v\}_{v\in V}, c_{uv}{}_{uv\in E}, c_{uv}^{coch}{}_{uv\in E}, c_{uv}^{adj}{}_{uv\in E})$. Formally:

Problem 1.3 *Given a pair of transmitters u, v with the corresponding edge uv we define the cost of a violation as:*

$$\varphi_{MI}(f, uv) = \begin{cases} c_{uv}^{hard} & if\ |f(u) - f(v)| < c_{uv} \\ c_{uv}^{adj} & if\ |f(u) - f(v)| = 1 \geq c_{uv} \\ c_{uv}^{coch} & if\ |f(u) - f(v)| = 0 = c_{uv} \\ 0 & if\ |f(u) - f(v)| \geq max\{c_{uv}, 2\} \end{cases}$$

Given an allocation of frequencies $F = \{1, 2, \ldots, K\}$ the Minimum Interference Frequency Assignment Problem (MI-FAP) *aims to produce an assignment $f : V \rightarrow F$ which respects the blocked channel constraints, that is $f(v) \in F\backslash B_v\ \ \forall v \in V$, does not violate any hard constraints and minimizes the soft constraints. This can* *be formulated as minimizing:*

$$O_{MI}(f) = \sum_{uv \in E} \varphi_{MI}(f, uv)$$

where only solutions with $O_{MI}(f) < c_{uv}^{hard}$ are valid.

In Problem 1.1 c_{uv}^{hard} is a large value chosen so that an assignment $\bar{f}$ with $O_{MI}(\bar{f}) \geq c_{uv}^{hard}$ is known to violate at least one of the hard constraints.

1.4 Other formulations and models

Although the three types of FAP described used for this thesis can be considered as the most important among those proposed with the binary model other formulations are possible. For example in their exhaustive survey recently published Aardal et al. also mentioned the Maximum Service FAP which can be seen as something intermediate between the MS-FAP and the FS-FAP. The problem is based on the fact that when feasible solutions are not available within the assigned spectrum we can seek for a partial solution that assigns as many frequencies as possible to the vertices. If we assume that for each node we have a fixed number of frequencies (services of variable bandwidth) available it will not be possible to assign all of them without interference and so some will remain unassigned. Hence the problem is formally defined as maximizing the number of frequencies that can be assigned (without interference) to each node in the network. A further development of this idea leads to the so called Minimum Blocking Frequency Assignment Problem which computes the actual blocking probabilities in the vertices as a function of the number of assigned frequencies. Here the objective function becomes a weighted combination of the blocking probabilities defined for each node in the network. More details about the formulations above can be find in [4].

The binary model described in Section 1.2 is the most concise representation generally adopted for the FAP, see [40, 60]. However, the main drawback of this model is that we are unable to represent more than one frequency for each transmitter. In case of cellular problems, in which each cell has to satisfy a given traffic demand d_v $\forall v \in V$, it may be useful to adopt other representations which have been proposed to overcome this problem. In [2] the function $f(v)$ is interpreted as a multivalued function where each element in its range is represented by a subset of $F \backslash B_v$. In [3] the FAP is described in terms of integer programming in the formulation commonly adopted to solve the FAP by exact methods such as tree search, see Section 2.1.

Alternatively, a different approach is constituted by the multiple interference model described in Section 2.2.4, which considers the interference produced by all the transmitters in the network when they transmit simultaneously.

1.5 Solving large FAP instances

The MS-FAP and its variants have been proven to be NP-Hard [53] by reduction to a graph coloring problem. Other simple reduction proofs have been provided to show that the other formulations of the problem are also NP-hard. Consequently, exact methods are able to solve the FAP only for small instances composed of a limited number of transmitters.

Successful solution techniques for the FAP are usually based on meta-heuristic algorithms, while lower bounding techniques have been developed that allow the quality of these solutions techniques to be assessed. This has been shown to be successful for some test problems. However, many of these meta-heuristics have not been usually demonstrated on very large scale problems that occur in practice or examples where tractability prospects are low. In these cases handling the whole assignment problem can be particularly challenging and good quality results may require considerable runtimes. Highly specialised algorithms are generally used, whose performance mainly depends on the specific problem for which they have been designed.

To resolve this, we propose a decomposed assignment approach to solve the FAP with meta-heuristics. Larger problems can then be handled and solved by meta-heuristic techniques within a shorter time.

Decomposed approaches have primarily been applied in combination with exact methods and very seldomly with heuristic techniques. In these cases authors usually proposed meta-heuristic algorithms which incorporate exact procedures for local optimization.

This thesis is the first work which investigates constructively this problem by describing a number of decomposition methods and their application to each type of FAP. Main focus is given to the MI-FAP but all the formulations proposed will be considered in order of difficulty. The FAP is solved by applying the decomposed approach to a number of meta-heuristics. Due to time restrictions we have limited the choice to a simulated annealing and a genetic algorithm thus including the different categories of evolutionary and local searches algorithms. In addition, the genetic algorithm has been implemented with two different assignment representations: the straightforward direct representation already used for the simulated annealing and the order-based, which has been previously used only for some of the formulations of the FAP (MI-FAP). Note that this representation requires the introduction of a further mapping performed by a sequential assignment thus including an example of constructive methods, which complete the classification of the most commonly used heuristics for the FAP. Furthermore sequential assignments are strictly related with the original idea of our decomposition approach which divides the network into subsets and then solves them sequentially (the two approaches actually coincide if we imagine a decomposition into $|V|$ subsets each composed of only one vertex)

Both heuristics have been applied following their standard procedure. Dealing with standard implementations adds generality to the decomposed assignment procedure proposed, whose effectiveness aims to be algorithm and problem independent. Moreover, the percentage of the improvement brought about in this case is expected to be more important than using non-standard high performing algorithms, thus the gap between different meta-heuristics can be reduced. In fact, the more sophisticated an algorithm is, the less are the advantages expected from the decomposition approach (which introduces approximations in the number of constraints considered). For example exacts algorithms are always expected to perform better with the global approach than with the decomposed technique (or in the the worst case equalize the results).

The results obtained for each type of FAP will then be presented and discussed. A number of partitioning methods are presented and compared for each FAP, such as clique detection; sequential orderings; novel applications of existing graph partitioning and clustering methods adapted for this problem.

Finally, all of the experiments performed for this thesis have been conducted on the resources provided by the Cardiff University Condor Pool [1]. This has allowed extensive simulations to complete the high number of required test. In order to make uniform the runtime values returned by the different machines in the pool a specific procedure has been implemented, whose detailed description is reported in Appendix B.

Chapter 2

Literature review

This chapter presents an overview of the techniques used to solve the FAP by classifying them in two main categories of exact methods and meta-heuristic algorithms. Subsequently, it reports a review of the previous works which have applied problem decomposition techniques to the FAP. Finally, the chapter concludes by describing briefly the main group of benchmarks used in the literature to solve this problem in its different formulations.

2.1 Exact methods and lower bounds

A number of lower bounds are available in the literature for the MS-FAP and are based, for the majority of cases, on its reformulation as a T-Coloring problem. They were firstly introduced in Gamst [44], which proposed a bound based on the clique of the interference graph, and successively by [112,114]. In addition, among the others, a second group of bounds is based on integer linear models either as a reformulation of the problem as a *vertex packing problem* [3] or on the *minimum Hamiltonian path* [8,12].

For the MI-FAP (and its generalization as FS-FAP) a more limited number of lower bounds have been proposed because of the difficulties introduced by considering some of the hard interference constraints as soft constraints which involves the use of penalty factors. In [66] a method based on the solution of a nonlinear problem, which is by construction a lower bound of the FAP, produced good lower bounds for some specific CELAR instances of the CALMA project which present an additional constraint called *mobility cost.* Koster et al. suggested in [74] a method based on the linear relaxation of a generalization of the FAP as a *partial constraint satisfaction problem* (see [72]). This produced some good results but only for instances with a very small frequency domain (less than three frequencies available for transmitter). In [82] the same MI-FAP formulation constitutes the starting point for deriving lower bounds based on analogies with the quadratic assignment problem. In [89] an improvement of the integer programming formulation, reinforced with additional constraints derived from the cliques of the interfering graph previously used in [88], has been used to produce lower bounds for different FS-FAP benchmarks. It is important to mention that this work presents an interesting attempt at applying it to larger MI-FAP instances of the COST-259 benchmarks. Subsequently, Smith et al. further improved these bound in a more recent work [108].

This work, together with [109] and [7], extends and update the result previously published in [8] and also summarizes and compare the performance in terms of lower bounds for the different formulation of the FAP. All of these papers improved the known lower bounds using methods from mathematical programming, which present some advantages towards the computation of lower bounds proposed before. In particular the calculation proposed in [8] is based on the detection of cliques which may be need to be eventually modified, and this can be computationally expensive, may require manual intervention, and its effectiveness strongly depends on the specific problem considered. On the contrary, the other methods based on the minimum Hamitlonian have the advantage of providing upper bounds as well but they are not always entirely successful. Moreover, their effectiveness may not scale well with the size of the data set. However, if the lower bounds generated for the MS-FAP are overall successful for real problems (but not for random generated problems [7]) the same conclusion cannot be drawn for the MI-FAP. In fact, for this type of FAP lower bounds are successful only in limited cases [7, 109] whereas for others (for example the COST259 benchmarks introduced later in Section 2.4) the gap between lower and upper bounds is still very large and so inefficient in practice.

Finally, Eisenblatter [40] derived new lower bounds for the COST 259 MI-FAP instances by studying the semidefinite programming relaxation of the minimum k-partition problem. These bounds are based on the fact that the MI-FAP reduces to a minimum k-partition problem, which can be modelled as a semidefinite program with the restriction of considering only co-channel interference.

Exact methods have been mainly proposed for problems for which lower bounds are available. The most common optimization method is *tree search* and a complete overview of this approach can be found in [4]. In tree search algorithms we can distinguish two main parts: the construction of the tree and the processing of its nodes. The first part consists of the choice of the variable for branching as well as the selection of a subproblem from the tree, usually using methods as *depth-first search* or *best-first search*. The second part concerns the actual process of solving a subproblem by applying reduction and node pruning techniques, such as cutting plane algorithms, and combinatorial lower bounding techniques. Most of the branching rules used for the FAP are static, i.e independent from the actual tree search, and consist only in selecting a vertex from an initial ordering usually based on the degree of the interference graph, such as *highest degree first* or *smallest degree last*. An outline of these ordering methods is given in Section 2.2.1. In [46] a dynamic selection called *saturation degree*, originally proposed in [17, 98] for the graph coloring problem, has been successively used to solve the MS-FAP with a branch-and-cut method. Similarly, in [3] the FAP is solved using another branch-and-cut algorithm which is tested on the smallest data sets of the CALMA project. However, these category of methods, as well as the branch-and-bound, has been reported to be usually successful only to determine whether or not a given assignment is feasible, despite requiring a large computational effort to solve the relaxation in each node of the enumeration tree. Alternatively, Mannino and Sassano proposed in [81] an exact enumerative method provided by pre-processing and fixing techniques to reduce the size of the instances while a restricted backtracking was used to reduce the size of the tree. The algorithm is used to solve MS-FAP instance only, although a cumulative interference is added to some of the instances in order to solve the FAP as a feasibility problem.

Very few attempts have been made to solve the more complex MI-FAP by exact algorithms. In [66] there are indications on the use of a branch-and-cut method but no computational results are provided. In [75] the formulation previously used in [74] to produce lower bounds has been extended to propose a tree decomposition algorithm (see also Section 2.3.1) but it has been proven to be successful only on instances presenting a particularly suitable structure. In [82] the lower bounds derived for the MI-FAP have been used, in addition to some of the the reduction and dominance rules reported in [112], to propose a branch-and-cut algorithm which has been tested on sub-instances of the CELAR data sets as well as some of the Philadelphia instances considered as fixed spectrum.

In conclusion, although partially successful for the smallest data sets, all the exact techniques proposed in the literature for the different models of FAP can be applied successfully only to relatively small data sets, usually in the range from ten to few hundreds transmitters, whereas the few attempts made on larger size data sets usually resulted in exceeding the maximum fixed CPU time constraint, as for example for the test reported in [82].

2.2 Meta-Heuristics for FAP

For practical instances the majority of research works have used heuristic approaches. We adopt in this thesis the classification proposed in [60] into *Constructive*, *Improvement* and *Evolutionary methods*.

Although heuristics have been demonstrated to perform reasonably well for small and medium size problems the assignments produced for large size ones can be far from the optimal, as reported in [111] for the MS-FAP. Furthermore, for the harder categories of FS\MI-FAP the have not generally been proven to be completely effective for every type of benchmarks, see for example [108] which provides limited evidence that meta-heuristics may not be fully effective for the MI-FAP by artificially constructing a number of benchmarks derived from the COST259 benchmarks (see [28]) for which the optimal solution is known.

2.2.1 Constructive methods

This first category consists of the *Sequential algorithms* proposed by Hale [54]. These were originally proposed for the MS-FAP and are the simplest and quickest methods to produce feasible solutions. They start with a sequential list of transmitters, ordered by some defined criteria. Frequencies are assigned to transmitters in turn, with the first transmitter receiving the lowest frequency. Subsequently, frequencies are chosen from those which do not violate any constraint with the transmitters already assigned in the ordering. Thus a feasible assignment is always guaranteed for a given ordering of transmitters. Because of their characteristic of progressively reducing the search space, the quality of their results is generally rather poor and depends heavily on both the initial chosen ordering and the particular problem considered.

In [54] the selection of the next transmitter is made according to a given *ordering* of the whole set of transmitters V. A number of different orderings are proposed including *Largest degree First* (LF) , *Smallest degree Last* (SL), *Generalized Largest First* (GLF), and *Generalized Smallest Last* (GSL). All of them aim to place the 'hardest' to assign transmitters at the start of the ordering. They all tend to produce non-ascending permutations with respect to the generalized degree of the vertices of the interfering graph. More sophisticated orderings are also proposed, such as the *Generalized Saturation Degree (GSD)* based on the saturation ordering previously introduced in Section 2.1.

Besides these procedures used for selecting the next transmitter other techniques are proposed in order to select and assign a channel to a given vertex. The simplest is termed the *Smallest Acceptable Frequency*, which selects the smallest available frequency among those that do not produce any violations. Other possible selections take into account the number of occupied channels, that is the set of frequencies already assigned to a transmitter, see *Smallest Acceptable Occupied* (SAO) and *Smallest Most Heavily Occupied* techniques (SAMHO).

Although in general they are not competitive with other methods in isolation, sequential algorithms can be used to produce effective upper bounds [17]. Moreover they can be used as preprocessing to produce starting assignments for other local search methods [62] or they can be incorporated in more complex algorithm structures. In Section 2.2.3 they constitute the evaluation procedure for an evolutionary algorithm. In [25] a variant of the procedure in [54] has been proposed to adapt the sequential assignment for the FS-FAP. In a more recent work [105] a new greedy algorithm has been proposed to solve MS-FAP instances. Although the authors stress the fact that the main advantage of their heuristic is to produce good approximations in computational time which increases only linearly to the number of transmitters, the method is capable of yielding optimum solutions to the majority of the Philadelphia instances tested. Finally, in a recent publication Chiarandini and Stutzle [20] propose new implementations of the sequential algorithm and compare them with the original formualtions in [54].

2.2.2 Improvement methods

Improvement methods are based on iterative local searches of the neighboring search space. *Local Search* (LS) is the most basic improving heuristic developed for combinatorial problems. An initial solution is selected and iteratively replaced with an improved one chosen from a restricted subset of similar solutions called a *neighbourhood*. A neighbour is obtained from the current solution by means of a given set of small changes called *moves*. For example, a 1-opt neighborhood is the set of solutions obtained from the current one by selecting a vertex and changing its frequency value (see Algorithm 7.4) whereas 2-opts are obtained by selecting two vertices and swapping their frequencies. A neighbour replaces the old solution if it produces a better value for the cost. Here the main issue is expanding the size of the neighbouring set without increasing too much the runtime of the algorithm. In fact, large neighborhoods correspond in general to exponentially increasing search times [4].

Because plain LS methods do not provide any mechanism for escape from local optima they have been very seldom used for the FAP and they must be applied many times with different random seeds and different initial solutions. In [62] *Hill Climbing* (see Definition 3.7) is used to solve FS-FAP instances while other examples of LS applied to the FAP can be found in [19, 97]. In these 1-opt and 2-opt neighborhoods are used and applied to a set of randomly generated instances of the MI-FAP.

Meta-heuristics which help LS to escape local optima are preferable choices for the FAP. In [115] a *Guided Local Search* applies variable penalty values to solutions trapped in local minima. This technique is applied to MI-FAP, MO-FAP and MS-FAP instances of the CALMA project. However, the most successful improvement methods for this combinatorial problem are *Tabu Search* (TS) and *Simulated Annealing* (SA), which are widely recognized high performing meta-heuristics for both the MS-FAP and FS-FAP.

SA was formally introduced by Kirkpatrick, Gelat and Vecchi for general optimization [70]. Subsequently, it has been adapted to solve variants of the FAP by different authors, with the energy function object of the optimization defined as a measure of the interference constraints. Notable applications of SA to the FAP are described in [13, 38, 100, 117]. Effective SA implementations can be found in [5] for the CALMA project instances, and in [64, 110] for other benchmarks. Pseudocode of a generic implementation can be found in [60] while that used in this thesis is outlined in Algorithm 7.1. In [57] a variant of SA, called *threshold accepting*, is applied to the COST259 MI-FAP instances with the difference of limiting the acceptable moves to configurations which respect the hard constraints. At the end of every loop the algorithm applies a so-called *one-cell optimization* obtained by letting all of the frequencies assigned to a vertex be changed simultaneously. [92–94, 102] propose bounds for the expected time needed by SA to find a optimal solution for different combinatorial problems, which include the graph colouring. In particular, they show a relation which is exponential with the size of the problem.

TS was originally defined by Glover in [47] and follows the basic idea of exploring the neighbouring spaces by a sequence of moves, in which a move is defined by the best available configuration. However, in order to escape from local minima, some moves based on the short-term and long-term history of the sequence of moves are classified as forbidden, or *tabu*, and stored in a *tabu list*. In more recent works TS has been extended to solve fixed spectrum FAP instances (see [15, 19]). In [19] the variant of TS proposed is called *Tabu Thresholding* and is implemented in two different phases called 'improving' and 'mixed'. In both of the phases the tabu list is substituted by a partition of the neighbourhing space into a number of subsets, which are then in turn further partitioned into blocks. Montemanni et al. improved in [87] the Tabu Search performance by proposing a new implementation which uses a dynamic tabu list and a cost function updating with a cost change table (allowing full neighbourhood) . Other TS implementations, solving both the MS and FS-FAP, can be found in [39, 62].

A recent comparison of the most effective local search algorithm for the MS-FAP can be found in [20] whereas [4] provides a complete overview for all the different formulations of the FAP.

2.2.3 Evolutionary algorithms

Despite the range of heuristic techniques that have been proposed for the FAP, it is still not easy to identify which methods can be the most effective on a wide range of test problems. This is particularly true for the category of evolutionary algorithms which includes the class of Genetic algorithms (GAs). These are search methods originally developed by Holland with the goal of either reproducing the natural process of evolution or adapting it to design software systems retaining its original mechanism. Pseudocode for a generic GA can be found in [58]. Pseudocode of the implementations used in this thesis are outlined in Algorithm 5.1 and 7.3.

On a closer investigation of the various implementations proposed, they appear to be problem specific and their effectiveness mainly depends on the particular data set considered whereas local search heuristics are more generally successful. In particular for the minimum interference instances of the harder FS-FAP there is no clear evidence of their general competitiveness. Although they adopt different representations to encode the population of chromosomes, they have very rarely been applied to more than one representation of the same instance for a given test problem, see [29, 65, 68], so comparisons are limited. Finally, some of the GAs proposed appear capable of producing good results but the instances used have not been made widely available to other researchers [31]. Three main categories can be identified:

Direct representation This first category encodes the chromosomes into either an integer vector or a set of integers [6, 21, 30, 31, 45, 55, 71] representing the frequency assigned to the corresponding transmitter. For the MS-FAP this representation allows solutions which are not feasible, that is solutions which violate some constraints.

Here the main difficulty arises in the choice of effective genetic operators since the standard ones produce poor performance. Some new problem specific operators have been suggested. They obtain the best performance on MO-FAP instance [55, 68] but the results obtained with the other types of FAP are not uniform, as observed in [4]. *Penalty factors* can be added to the fitness function in order to weight constraints in a different way, but the setting of the weight values, as well as the other parameters used, is rather a difficult task, as noticed by the same authors. In [31, 65, 78] some good results were obtained for the MI-FAP but the problems tested (although represent real-life instances) do not belong to any of the benchmarks widely available [2] and they either have not been compared with other methods or the benchmark used is not shown. It is worth noticing that with this representation it is particularly difficult to find a crossover operator able to transmit good properties to children solutions without being too disruptive. In some works the GA is implemented with no crossover applied, which essentially results in a local neighbourhood search [55, 68, 106].

In [29, 30] this representation is extended to test some of the FS-FAP problems of the CALMA project and other real world instances. Here, the problem is approached in two phases: the first optimization step aims to find a feasible solution by minimizing the number of violated interference constraints whereas the second searches for a solution with minimal interference costs, while keeping the solution feasible. In [71] an innovative approach produces very good results on one of the widely available benchmark of the CALMA problems. However the genetic operators used are 'optimal' operators whose application is computationally very expensive as observed in [4, 79]. In particular the crossover implements a branch and cut procedure, which also makes the GA a hybrid algorithm as discussed later in this thesis.

A variation of this representation was firstly proposed in [65] and found superior to the above direct representation. Here chromosomes are encoded in subsets (genes) which include the vertices which are assigned the same frequency. However, the GA proposed involves the use of parallel computing and it has been outperformed by other heuristics, such as SA, as reported in a later work [64]. Few other works adopt the same representation but the results obtained are essentially similar to those produced by the direct one (see [99]).

Bit String Representation The second category uses a similar approach but encodes the chromosomes into a binary string [91, 107]. Although the cost function is still set as a measure of the total violations, this representation can generate solutions which automatically satisfy some of the constraints, such as co-cell constraints, thus reducing the search space and improving the computational efficiency of the GA.

Permutation Based Representation The third type of representation, which was originally proposed for the MS-FAP in [118, 119] and independently in [14], is the 'permutation-based' representation (also known as 'order-based'). Genes are integers representing transmitters, and individuals are represented by permutations of the set of integers including all the transmitters. The fitness of an individual is then produced by a sequential algorithm to assign frequencies to transmitters thus producing at each evaluation a feasible solution for the MS-FAP. In the original work [118] a simple steady-state GA produced excellent results on widely accepted minimum span test data sets including the Philadelphia instances. The latter set of benchmarks has been also used in [14] with a generational based GA using the same representation.

In more recent work an adaptation of the sequential assignment for the FS-FAP has been used as evaluation procedure for the steady-state algorithm proposed in [25] and tested on a number of FS-FAP benchmarks. However the use of this representation does not guarantee the complete coverage of the search space. Therefore a local search procedure has been incorporated into the GA structure.

2.2.4 Shortcomings of other existing approaches

The technique of hybridizing GAs with another heuristic in order to improve their performance forms the class of *memetic algorithms*. This technique is often used for the more general category of constraint satisfaction problems, which includes the FAP, as described in [79]. In this broader category the algorithm often incorporates a LS in the GA structure in order to either act as a repair mechanism when the genetic operators are highly disruptive [31] or to diversify the search to increase solution quality when the algorithm is trapped in local optima [79,91]. Often the GA is used only to produce good approximations which are subsequently improved by a local heuristic [6, 83, 99]. In a completely opposite approach, the GA can act as second operator after other methods, for example stochastic ranking [123], simulated annealing [116], and neural networks [101].

There is a recognized need to elaborate more sophisticated interference models, which can solve more realistic frequency assignment problems, as described in [36, 38, 110, 124]. It would be more accurate to take into consideration the unwanted signal caused by all transmitters when they transmit simultaneously. *Multiple interference* models can be found in the following works. Lower bounds using a multiple interference model are proposed in [108]. In [76] interference constraints are divided in hard and soft constraints, and furthermore in co-channel and adjacent-channel, each of them assigned to a different set of transmitters. [110] proposes a direct multiple interference model beside an intermediate model, which still uses binary constraints to represent multiple interference. These two models have been adopted to solve MS-FAP instances using simulated annealing and multi-agents algorithms (ANTS) [86]. Other slightly different multiple interference models can be found in [42, 81]. However, there is a general lack of results about models which differ from the binary, and no results are actually available for the FS-FAP instances. In addition, multiple interference becomes a fundamental issue in the design of real systems like infrastructure and independent, or ad-hoc, wireless networks, see for example [121].

Finally, [61], [18], and [52] give further important contributions to the application of multiple interference models. Here given a series of reception points the evaluation of an assignment requires that for each of these points the ratio between the receiver power and the sum of powers received from interfering transmissions (known as *signal-to-interference* ratio) is above an assigned threshold. Nearly optimal asignments are then produced by different local search heuristics, such as tabu search [18] and simulated annealing [61], in which the cost function objective of minimization is the sum of the signal-to-interference ratios over all the transmitters in the network. [52] shows that (using SA but the conclusions achieved can be extended to other meta-heuristics) further improvements are obtained if the cost function becomes a combination of that used for the multiple interference and the binary constraint model. This can be reaches in two ways either by the addition of a term which measure the constraint violations (as the interference modeled by the binary constraints graph) or by using the binary model to produce a starting solution to be used as input for the multiple interference implementation.

However, the main contribution of these paper is that they all show that the use of more realistic models, such as the multiple interference one, produces benefits in either the production or the evaluation of frequency assignments which are far more important than small improvements to artificial cost functions.

Finally, since the main argument against the use of these models is about their greater resources required, the decomposition approach object of this thesis has the potentiality of being profitably applied to multiple interference models which minimize the signal-to-interference ratios directly.

2.2.5 Summary of meta-heuristics approaches

To summarize, the FAP in its different formulations have been more commonly solved by a heuristic approach. However, although meta-heuristics produce good results on some of the benchmarks available, highly specialised algorithms tend to perform best. In addition, standard implementations of meta-heuristics may require considerable runtimes to produce good quality results whenever a problem is very large or complex.

This thesis investigates the application of problem decomposition techniques as a possible solution to this drawback. Problem decomposition can also be thought of as an alternative to the introduction of exact procedures to optimize the heuristic solutions which inevitably increases the complexity of these algorithm thus limiting the range of their applicability.

2.3 Problem decomposition for FAP

Previously published works which have applied decomposition for the FAP, can be grouped into three main categories; used in combination with exact methods; with meta-heuristics as a second phase optimization (either following or incorporated the heuristic procedure); and the approach proposed in this thesis based on constructively finding an efficient decomposition into subproblems that leads to corresponding partial solutions. These will then be recomposed into a solution of the initial problem.

2.3.1 Decomposition combined with exact methods

The most common application of problem decomposition techniques for the FAP has been with exact methods. We have described in Section 2.1 how a number of them are based on selecting an initial ordering aiming to consider any possible hard part of the data set first. In [81] this idea is further developed by identifying a hard subgraph, called the *core*, which is isolated and solved first. Then the remaining part of the problem can be solved without ideally influencing the global objective function. In Aardal et al. [3] a preprocessing phase based on the cliques of the interference graph is used with a branch-and-cut algorithm to reduce the size of minimum order problems (MO-FAP) by between ten to fifteen percent. Similarly, a clique bound has been extended and generalized to provide lower bounds for both the MO-FAP and MS-FAP.

Koster et al. [75] observed that assigning frequencies to a cut-set of the interference graph decomposes the problem into two or more independent subproblems, thus they generated a sequence of such cut-sets using tree decomposition. This idea, in addition to the use of several further dominance and bounding techniques led to the solution of some small and medium size instances for the MI-FAP. However, for larger real-life instances in which the proposed dynamic programming algorithm is impractical because of the width of the tree, the algorithm has been used iteratively to improve some known lower bounds. However, the methods described above can produce solutions in a reasonable run time only for the easiest instances. As a consequence they have been primarily used either to produce lower bounds or as a preprocessing technique.

2.3.2 Decomposition combined with heuristic methods

Cell re-optimization

Decomposition has rarely been used in combination with meta-heuristics. In these cases a decomposition of the whole set of transmitters into a number of subsets has been used to optimize the solution either after the heuristic method or during its procedure at a fixed number of iterations. Moreover, the partitioning adopted is similar to that used for distributed channel assignment for cellular problems in order to optimize solutions locally, usually by applying an exact procedure, inside system clusters of several cells [51, 67, 122].

Hellebrandt and Heller proposed a *cell re-optimization* method used in combination with their so called *Threshold Accepting* method [57]. The procedure consisted of optimizing each cell assignment after a fixed number of iterations. Each cell is selected in sequential order and its assignment is (re)optimized by an exact method, while those in the other cells are kept fixed. These results were at the top of the list of the COST259 test problems at the time they were published. Subsequently Montemanni et al. used the same cell implementation procedure to improve their results obtained with Tabu Search [87] (which also improved those of Hellebrandt and Heller).

In a similar fashion Mannino et al. [80] proposed a new sophisticated implementation of SA combined with dynamic programming to compute local optima. Their method applies the original re-optimization idea to a cluster of cells. In their approach, they optimize assignments in cliques of vertices of multiple demand by reducing this problem to finding fixed cardinality stable sets in interval graphs. Here a modification of SA, in which the neighborhood consists of only the configurations satisfying the hard constraints, is combined with a *re-optimization* performed at the end of every loop of the algorithm. The authors shown how a current best solution obtained by SA could be optimised by letting all of the frequencies assigned to a vertex be changed simultaneously. They also shown that this corresponds to looking for a minimum cost k-cardinality stable set in interval graphs, where k is the demand of the vertex.

In these examples the local optimization procedure obtains very good results on the COST259 MI-FAP instances (overtooking some of the best results published so far, see [2]), although the addition of elaborate exact procedures considerably increases the computational complexity of the algorithm thus requiring runtimes roughly one order of magnitude higher than those of the fast heuristic combinations [40].

Subgraphs

In this thesis the decomposition strategy is extended to a larger scale by adopting a different approach in which problem decomposition is used to divide a complex FAP instance into a number of subproblems, which can be then more effectively solved and recomposed into a solution of the original problem. It is worth noting that this approach does not involve any exact local optimization algorithm and therefore it is suitable for the application of standard algorithms. Here the decomposition approach aims to simplify a complicated problem by considering separate subproblems obtained by removing some of the constraints between pairs of vertices representing transmitters, rather than increasing the algorithm complexity and solving the problem as a whole.

This approach has seldom been used in the literature. In [111] Hurley et al. extended the standard and generalized clique bounds originally proposed for the MS-FAP and MO-FAP (see for example [44]) to a heuristic approach. They start by finding a level-p clique, which is the largest clique having minimum weight edges of p, then they produce a first assignment for the clique by applying a meta-heuristic and evaluate its span. Subsequently, the clique assignment is kept fixed and an attempt is made to extend the assignment to the full interference graph. If the span of the final assignment is not close to that of the clique, a number of vertices are added to the clique, creating a so-called *near clique*, and the process is repeated until the difference between these two span values is below a given threshold. This procedure produced good results on some MS-FAP instances including some of the Philadelphia benchmarks and other test problems provided by Cardiff University (see [19]). However, it presents the drawback of finding the maximum cliques in the graph, which is an NP-hard problem itself.

Note that an idea similar to subgraphs have been used with exact methods when a subset o the vertices is firstly assigned ann then this assignment being extended to a complete one, see the approach used in [81] described in Section 2.3.1.

A slightly different approach has been recently used in [69]. Here a clustering algorithm based on the generalized-degree of the neighbour vertices is used to initially partition a real-life FS-FAP instance, which is then solved by a GA. However, the representation used has the limitation of only considering co-channel interference. Moreover the elements included in different subsets exchange their position during the optimization process, which makes this algorithm considerably different from our decomposition procedure.

Finally, in [25] an order-based steady-state *genetic algorithm* (GA) has been combined with two different decompositions, based on either the generalized degree of the corresponding graph or more sophisticated graph partitioning algorithms, to solve both the MS-FAP and simple instances of the FS-FAP. Furthermore, [26] presents preliminary results of applying the same procedure to the MI-FAP using a generational GA with direct representation. In the rest of the thesis we will generalise the use of this approach to all the types of FAP defined in Section 1.2 and extend it to a wider range of decomposition methods.

2.4 Benchmarks for FAP

This section outlines briefly the benchmarks most commonly used for the FAP. We will limit the description to those publicly available.

Philadelphia The Philadelphia test problems were one of the first benchmarks proposed for the MS-FAP. Network sites were modeled on a hexagonal grid and each of them demands a high number of frequencies equal to the multiplicity of the sites. In the first instance transmitters belonging to the same site or adjacent sites cannot use the same frequencies. However, different formulations are proposed which introduce the concept of *re-use distance* [10]. While not realistic in practical terms, these benchmarks are widely quoted.

Calma The CALMA instances represent military applications and differ from other frequency assignment problems by their specific distance separation constraints. Besides the minimum distance constraints they also present *equality* constraints, that is two frequencies at a fixed distance must be assigned to the corresponding vertices, and *mobility* constraints, which penalizes changes in some fixed frequency values assigned to specific transmitters in the network.

Cardiff University These instances are thought to simulate more realistic wireless network and are divided in two groups. The first group was generated by a specific tool which locates transmitters according to a given probabilistic distribution [9] and aim to produce larger benchmarks than those provided in the literature, whereas the second group includes real *global system for mobile communications* (GSM) scenarios. Note that for very large benchmarks the need for problem decomposition techniques becomes compulsory since standard meta-heuristics are in general not able to produce competitive results while more sophisticated ones cannot be actually applied because of their computational complexity.

COST-259 The COST 259 project on Wireless Flexible Personalized Communications ran in the second half of the nineties and consisted of many research group each of them working on different aspects of radio mobile communication, such as systems, antennas and propagation, and networking (see [2]). The outcome of this work was the constitution of a library of GSM frequency planning scenarios with the aim of producing new benchmarks for an updated comparisons of available frequency planning methods as well as the development of new ones. As a consequence, the scenarios proposed were rather different, although they all present the common characteristic of implementing the minimum interference model for the FAP. The final report of the COST 259 project has been finally published in [28]. These data sets have been explicitly designed and used to solve MI-FAP instances.

Random Graphs A $G_{n,p}$ random graph is defined as a graph of n vertices, such that the probability that any two given nodes are connected by an edge is p, independently for each pair of nodes. These are known to be very hard problems, although they do not represent any real network, since they use only one parameter to model the entire network and this is independent of the geometry of the configuration and correlations among links [41]. Moreover, they generally present a very high graph density which is not generally suitable for the application of decomposition techniques. However, a few of these instances will be included as a comparison between different decomposition methods for the two main types of FAP considered, that is MS-FAP and FS-FAP.

Chapter 3

Decomposition and assignment algorithms

This chapter will introduce our proposed *decomposed assignment procedure* for the FAP and, subsequently, will outline the different algorithms used to obtain such decomposition and the meta-heuristics used to produce the assignment solution of the problem.

3.1 Decomposed assignment approach

The procedure starts by partitioning the interference graph into one or more subsets. Then a meta-heuristic is applied to each of the subsets in turn to produce a sequence of partial solutions. When the current subset is considered, the algorithm keeps the assignment of transmitters in the previously assigned subsets fixed, and minimizes the constraint violations with them. Finally, the algorithm returns a final assignment, a solution of the whole problem.

Pseudocode of the decomposed assignment procedure is outlined in Algorithm 3.1. Note that it is stated to be applied to any MS/FS/MI-FAP. We need to distinguish between the first assignment loop over the subsets in the partition and the further ones. The first loop builds a sequence of partial assignments in which some vertices are unassigned and not considered in the cost function. At the end of this first loop, a complete assignment is obtained and subsequently the algorithm changes the assignment of a single subset during each iteration.

Definition 3.1 *Given a partition of V into $nSubs$ subsets* $\{ V_1, V_2, \ldots, V_{nSubs} \}$ *we define the set of the* intra *and* inter-edges *for a given subset V_j as*

$$E_j^{intra} = E(G[V_j]) \quad E_j^{inter} = \{uv : \ u \in V_j, \ v \notin V_j, \ uv \in E \}$$

In Definition 3.1 $G[V_j]$ $V_j \subseteq V$ indicates the subgraph induced by a subset V_j of V. For some purposes the subsets are solved independently, and this is shown in Algorithm 3.2. This algorithm builds distinct partial assignments for each of the

Algorithm 3.1 Decomposed assignment

```
Input:   $G(V, E)$, number of loops $nLoops$, size of partition $nSubs$
Output   FrequencyAssignment $f$ of V
1: Produce a partition { $V_1, V_2, \ldots, V_{nSubs}$ } of $V$ using decomposition algorithms
2: for $j = 1$ to $nSubs$ do // First loop
3:    Apply a meta-heuristic to determine $f(v)\ \forall v \in V_j$ to minimize the cost
         $$O_{MS/FS/MI}(f) = \sum_{uv \in E_j'} \varphi_{MS/FS/MI}(f, uv)$$
4:    where $E_j' = (E_j^{inter} \cup E_j^{intra}) \cap E(G[V_1 \cup V_2 \cup \ldots \cup V_j])$
5: end for
6: for $i = 2$ to $nLoops$ do
7:    for $j = 1$ to $nSubs$ do
8:       Apply a meta-heuristic to determine $f(v)\ \forall v \in V_j$ to minimize the cost
            $$O_{MS/FS/MI}(f) = \sum_{uv \in E_j} \varphi_{MS/FS/MI}(f, uv)$$
9:       where $E_j = (E_j^{inter} \cup E_j^{intra})$
10:   end for
11: end for
```

subsets. Further loops other than the first lose significance since during a partial assignment the procedure only considers the internal edges E_j^{intra} of the current subset j. Note that only the vertices included in the current subset are considered during its channel assignment. Finally, all the partial assignments produced are recombined to generate a final assignment of the original problem. Since the subsets are solved independently they can be solved in parallel, thus reducing the computational time required proportionally to the number of subsets.

As an example, we apply the decomposition procedures described above to the simple graph of ten vertices in Figure 3.1. Let $V_1 = \{v_1, v_2, v_3, v_4\}$, $V_2 = \{v_5, v_6, v_7\}$, $V_3 = \{v_8, v_9, v_{10}\}$ be a partitioning of the graph into three subsets. We define the following intra-edges $E_j^{intra} \subset E$ as $E_1^{intra} = \{v_1v_2, , v_1v_3, , v_1v_4, , v_2v_3\}$, $E_2^{intra} = \{v_5v_6, , v_5v_7, , v_6v_7\}$, and $E_3^{intra} = \{v_8v_9, , v_8v_9, , v_9v_{10}\}$. Let then $Ej^{inter} \subset E$ be the inter-egdes of subset V_1 with the other subsets; we can then define them as $E_1^{inter} = E_{12}^{inter} \cup E_{13}^{inter}$ where $E_{12}^{inter} = \{v_1v_5, v_3v_5, v_3v_7, v_4v_5, v_4v_6\}$ and $E_{13}^{inter} =$

Algorithm 3.2 Decomposed assignment - subsets solved independently

```
Input:    $G(V, E)$, number of loops $nLoops$, size of partition $nSubs$
Output    FrequencyAssignment $f$ of V
1: Produce a partition { $V_1$, $V_2, \ldots, V_{nSubs}$ } of $V$ using decomposition algorithms
2: for $j$ = 1 to $nSubs$ do
3:     Apply a meta-heuristic to determine $f(v)$ $\forall v \in V_j$ to minimize the cost
           $O_{MS/FS/MI}(f) = \sum_{uv \in E_j^{intra}} \varphi_{MS/FS/MI}(f, uv)$
   end for
```

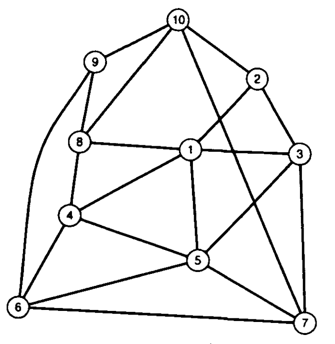


Figure 3.1: Example of binary constraints graph with ten vertices

$\{v_1v_8,\ v_2v_{10},\ v_4v_8\ \}$ are the inter-edges that V_1 has with V_2 and V_3 respectively. Similarly $E_2^{inter} = E_{21}^{inter} \cup E_{23}^{inter}$ where $E_{21}^{inter} = E_{12}^{inter}$ and $E_{23}^{inter} = \{v_6v_9,\ v_7v_{10}\}$. Finally, $E_3^{inter} = E_{31}^{inter} \cup E_{32}^{inter}$ where $E_{31}^{inter} = E_{13}^{inter}$ and $E_{32}^{inter} = E_{23}^{inter}$.

We describe the sequential procedure in Algorithm 3.1 first. The first loop starts by considering the first subset V_1 and producing a partial assignment of its vertices v_1, v_2, v_3, v_4 by applying a generic meta-heuristic. Note that its type and representation, as well as the procedure used to produce the assignment, do not need to be specified in this context. Only the intra-edges E_1^{intra} of V_1 are here considered. Note that the vertices belonging to the other subsets V_2 and V_3 remain unassigned in this phase. Then we consider the second subset V_2 ad we produce an assignment of its vertices. In this phase the assignments of the vertices of subset V_1 remain fixed whereas those of subset V_3 are still unassigned. We now consider the constraints represented by the intra-edges E_2^{intra} but only inter-egdes between the subsets already assigned, i.e. the set E_{21}^{inter} between subsets V_1 and V_2. To complete the second loop we now consider the last subset and we assign its vertices v_8, v_9, and v_{10} keeping fixed the assignments of all the other vertices. We here consider the whole set of inter-edges $E_3^{inter} = E_{31}^{inter} \cup E_{32}^{inter}$ since all the other subsets have been assigned. Note that at the end of this loop we have produced a complete assignment of all the vertices of the graph, simply obtained by concatenating the assignments produced for each subset.

If we conduct an further loop through the subsets we are now changing the assignments of the vertices included in the current subset only, whereas those of all the other vertices are kept fixed. This phase will always consider the intra-edges and the whole set inter-edges for each of the subsets considered. Namely, for V_1 we will consider $E_1^{intra} \cup E_1^{inter} = E_1^{intra} \cup E_{12}^{inter} \cup E_{13}^{inter}$. Then for V_2 we will consider $E_2^{intra} \cup E_2^{inter}$ and for V_3 $E_3^{intra} \cup E_3^{inter}$. At the end of each re-assignment of the vertices of the subsets currently examined we always have a complete assignment simply obtained by concatenating the assignments so far produced for each of the subsets.

When we apply the procedure in Algorithm 3.2 the subsets are solved independently and only one loop is considered. Moreover, for each subset we only consider the constraints represented by its intra-edges. For example, when we are solving subset V_1 we produce an assignment of its vertices v_1, v_2, v_3, v_4 considering only E_1^{intra} in the same way we have proceeded with the sequential Algorithm 3.1. Alternatively, instead of the direct assignment of the vertices the solution returned may be encoded into a specific representation required by the meta-heuristic. Subsequently, the same procedure is used for the next subset V_2. Note that since we only consider its intra-edges E_2^{intra} it does not matter whether or not we are keeping fixed the vertices already assigned, i.e. those of subset V_1 (actually the subsets could be solved in parallel with further gain in runtime terms). At the end of the application of the meta-heuristics to V_2 a partial assignment of its vertices v_5, v_6, v_7 (or an encoded solution) is returned. In a similar way we operate with V_3. At the end of the last subset a complete assignment of all the vertices of the graph is obtained either by simply concatenating the assignments produced for each subset or by applying a decoding procedure to the final solution returned (still obtained by concatenating the single solutions produced for each subset) if a specific representation of the solution is required by the meta-heuristic used.

3.2 Decomposition algorithms

This section describes the different decomposition methods that have been tested in order to compare their performance when the corresponding partitions are used to solve the FAP. In the algorithms outlined in the following we will refer to the weighted unordered graph $G(V, E)$ as the interference graph representing the network. However, in the case of the MI-FAP the contemporary presence of different types of constraints requires the introduction of an equivalent set of edges to reduce the network to a weighted simple graph G^D that combines the hard and soft constraints. For these instances G^D will actually replace $G(V, E)$ in all the decomposition algorithms here proposed.

Definition 3.2 *Given the interference graph $G(V, E)$ and the 3-tuple representing the constraints $({c_{uv}^{hard}}_{uv\in E}, {c_{uv}^{coch}}_{uv\in E}, {c_{uv}^{adj}}_{uv\in E})$ we define the graph $G^D(V, E)$ as the unordered binary graph having the vertex set V and edge set E with the edge weights given by the linear combination:*

$$c_{uv} = max\{\lambda_1 c_{uv}^{coch} + \lambda_2 c_{uv}^{adj}, \lambda_3 c_{uv}^{hard}\}$$

in which λ_i are assigned weights to reflect the relative importance of the constraints.

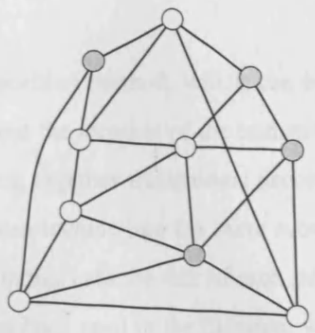


Figure 3.2: Example of random decomposition into two subsets for a graph with ten vertices

3.2.1 Random

Random decomposition is the simplest decomposition method. The partitioning produced can have subsets of either equal size or, in its general definition, random size. Note that this method uses no information about the distribution of transmitters or constraints within the network. Pseudocode is outlined in Algorithm 3.3.

Algorithm 3.3 Random decomposition

```
Input:   G(V,E), number of subsets $nSubs$, sizes of partitions $size_j$
Output:  Partition { $V_1, V_2, \ldots, V_{nSubs}$ } of V

1: Let $P = \{1, 2, \ldots, nSubs\}$
2: for $j = 1$ to $nSubs$ do
3:     for $l = 1$ to $size_j$ do
4:         Select at random an integer $k \in P$
5:         Assign the vertex $v_k \in V(G)$ to the subset $V_j$
6:         Remove $k$ from the set $P$
7:     end for
8: end for
```

Figure 3.2 shows an example of random decomposition applied to the graph in Figure 3.1. To simplify we only consider a decomposition into two subsets whose vertices are represented by white and shaded nodes respectively.

3.2.2 Geographical

A second simple decomposition method, which can be adopted only when geographical information about the location of the transmitters in the network is provided, consists of grouping together transmitters according to a geographical criterion. For instance, we can include into the same subset all transmitters within a fixed distance. Note that in this case the size of each subset will be determined.

A similar criterion has been used in the literature with the *distributed channel assignment*. In particular, these techniques are primarily used with cellular networks and, as a consequence, very often the decomposition adopted to produce assignments which consist of clusters of one or more geographically close cells, see [51,67,122]

This decomposition methods still ignores any information about the number/size of constraints between pairs of subsets. However, it may be able to produce satisfactory results when clusters of transmitters are concentrated around specific areas of the network, to whom we can refer as 'towns'. Note that, this is (to the author's knowledge) the procedure used in real applications by the network operators when they divided it into smaller areas. The algorithm used in this paper for geographical decomposition is outlined in Algorithm 3.4. An example of its application to the graph in Figure 3.1 is given in Figure 3.3.

Algorithm 3.4 Geographical decomposition

```
   Input:    $G(V, E)$, number of subsets $nSubs$, distance $D$
   Output:   Partition { $V_1$, $V_2, \ldots, V_{nSubs}$ } of V
1: while $V(G) \neq \emptyset$ do
2:     Select a vertex $v_k \in V(G)$ at random and include it in subset $V_j$
3:     Add to $V_j$ all the vertices $u$ for which $d = |u - v_k| \leq D \quad u \in V(G)$
4:     Remove all the vertices included in $V_j$ from the set $V(G)$
5: end while
```

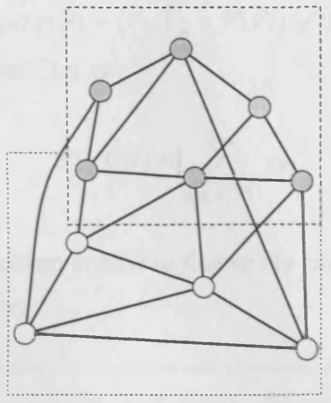


Figure 3.3: Example of geographical decomposition into two subsets for a graph with ten vertices

3.2.3 Minimum-cut

This simple decomposition criterion takes into consideration the structure of the interference graph representing the network, based on the *minimum-cut* algorithm. The idea, also used with exact approaches (see [75]), is that assigning frequencies to a *vertex cut* of the interference graph decomposes the problem into two independent subproblems. Hence, if the cut found is an empty set the two subproblems can be solved separately by the procedure in Algorithm 3.2 without any loss of quality when compared to the original problem. Consequently, because the heuristic is now solving smaller and easier subproblems than the problem as a whole based on the entire graph G, its performance is expected to improve considerably.

If we formulate the problem in terms of edges instead of vertices we are then looking for the smallest subset of edges whose deletion will disconnect the interference graph. Similarly, if we find an empty set the problem can be exactly reformulated as two independent subproblems. Alternatively, the quality of the solution produced by the decomposed approach can depend on the cost of the cut-set. The higher this cost, the less optimal the solution produced.

Problem 3.1 *Given a cut $c(G) = (V_1, V_2 = V \backslash V_1)$ of the unordered weighted graph G we will define the cost $C(c)$ as*

$$C(c) = \sum_{uv \in E_1^{inter}} c_{uv}$$

The minimum-cut problem *consist in finding the cut of the graph with minimum cost C for all cuts c of G.*

There are many algorithms proposed in the literature which solve the *minimum-cut* problem. Although traditional approaches use flow techniques and formulate the problem as a *minimum-cut maximum-flow* [59], in this thesis we have applied a simple fast deterministic non-flow algorithm proposed by Stoer and Wagner [113], which is based on *maximum adjacency search* methods.

Although this decomposition method is expected to produce better results than those presented so far since it considers the actual cost of the cut between the two subsets, it has the drawback of usually funding cuts with very unbalanced sizes, that is one of the sides of the cut composed by only few transmitters. Moreover, the size of smallest subset produced it is usually very small for the most of the FAP benchmark tested. A possible remedy is to apply the minimum-cut procedure iteratively by removing from the graph G the vertices included in the smallest subset of the cut, and then continuing until we obtain the minimum required cardinality for each part of the cut. Finally, the procedure needs to be adapted in order to obtain a number of subsets greater than two. This can be obtained by applying the procedure recursively to each part of the cut produced.

As example Figure 3.4 shows a minimum cut decomposition into two subsets for a graph with ten vertices reproduced from in [59]. To simplify we consider only unit weights of the constraints.

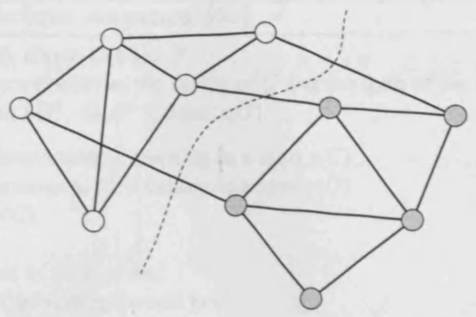


Figure 3.4: Example of minimum cut partitioning into two subsets for a graph with ten vertices

3.2.4 Cliques

The most important method which applies the subgraph approach (see Section 2.3.2) previously used in combination with meta-heuristics is clique decomposition. In [111] some MS-FAP instances were solved by a standard simulated annealing algorithm in two or more steps which involved partial assignments generated by solving the frequency assignment problem on the subgraph induced by the largest level zero clique and then extending it by adding some more vertices until a complete assignment for all the transmitters in the network is eventually reached. The addition of the remaining vertices was either made in a single step or through intermediate steps depending on the cost difference between the partial and final solutions obtained. If this gap is too big more vertices are added to the subset and the procedure starts again producing a new ‘extended’ partial assignment. Pseudocode of this procedure is outlined in Algorithm 3.5.

The maximum clique algorithm implemented in [62] was that proposed in [27], which provides an exact recursive method for finding the maximum clique of an unweighted graph. However, since for the MS-FAP the interference graph is weighted, with the weights representing the required channel separation which guarantees interference free assignments, the algorithm was modified in order to be able to find cliques at different levels. In the procedure proposed, level values

Algorithm 3.5 Subclique assignment [62]

```
Input:   $G(V, E)$, clique $C(G) \subseteq V$
         Tolerance $t$ between the span $s$ of $G$ and the span of the subgraph $G^S \subseteq G$
Output:  Partition { $G^S$, $G \backslash G^S$ }, Span $s(G)$

1: Assign $C$ meta-heuristically, resulting in a span $s(C)$
2: Extend the assignment to $G$, resulting in a span $s(G)$
3: $\sigma \leftarrow s(G) - s(C)$
4: $\sigma_{old} \leftarrow \sigma$
5: while $\sigma > t$ and $\sigma < \sigma_{old}$ do
6:     Select a set $W$ of vertices to add to $C$
7:     $C \leftarrow G[V(C) \cup W]$
8:     Assign $C$ meta-heuristically, resulting in a span $s(C)$
9:     Extend the assignment to $G$, resulting in a span $s(G)$
10:    $\sigma_{old} \leftarrow \sigma$
11:    $\sigma \leftarrow s(G) - s(C)$
12: end while
13: $G^S \leftarrow C$
```

represented the minimum weight of the edges in the clique, so *level zero* actually corresponded to the unweighted solution, level one to the clique composed by vertices requiring at least one channel separation and so on.

In order to deal with the different kind of weights that occur in the MI-FAP we have also implemented the weighted version of the algorithm proposed in [96], in which the pruning condition has been modified to be suitable for generic weighted graphs. The algorithm starts by ordering the weights in a given order, usually decreasing by general degree. In our implementation we adopted the GSL ordering proposed in [54] which gave the best results in [62]. Since the maximum clique problem is NP-hard, finding a good ordering plays a crucial role in order to reduce the runtime of the algorithm or the best result if the algorithm is terminated before completion (in our case, good approximations of the actual maximum clique may also be acceptable). However, for sparse problems the algorithm is faster without the use of any ordering [96].

An important concept in the algorithm is the *depth* of the search. At depth zero all the vertices are considered. Subsequently, the algorithm expands one vertex at a time (according to the given initial ordering), with the expansion operator consisting of listing all the adjacent vertices already included in the previous depth. The resulting list constitutes the next depth, then selection and expansion are repeated until no adjacent vertices are found. Finally the algorithm returns iteratively all the selected vertices which, because of being adjacent to each other, form a clique. If we repeat the process for every vertex, and we store and update the largest clique found so far, then the maximum clique will be found. To speed up the process a *pruning* condition is introduced. If at the current depth no cliques larger than the current best can be found the procedure returns. However, if we consider weighted graphs this condition needs to be modified.

Firstly, each vertex V_i is given an associated weight w_i equal to the sum of the weights c_{uv} of all its incident edges. The initial vertex ordering will be descending by vertex weights. Let d be the current depth, i the index of the currently selected vertex at depth d, and w_{di} the vertex weights which still need to be expanded. If the weight of the current clique plus the weight of the remaining vertices at the current depth is less or equal to the weight of the current largest clique found, the algorithm will prune.

Although faster modifications of the algorithm have been recently introduced (see [95]), because of the particular selection ordering used the method considered here, can be still effectively used to find good approximations of the maximum clique in a reasonable time, even for the largest instances tested . Pseudocode of the recursive maximum clique subroutine procedure, for both the weighted and unweighted versions, is outlined in Algorithms 3.6 and 3.7, where $N(v_i)$ represent the set of vertices neighbours of vertex v_i (see also [95]). At the first call of the procedure the set of vertices U will coincide with the entire set $V(G)$ of the vertices of the graph. Note that in the algorithm proposed for the unweighted version [96] the graph considered is weighted on its vertices instead of the edges. To overcome this we introduce an artificial modification to the interference graph G in order to include all information about weights within the vertices.

Definition 3.3 *In order to apply the maximum clique algorithms [27, 96], given the unordered weighted graph G^D representing the network we consider the following* auxiliary graph G^C *such that:*

- for the unweighted version: $G^C \equiv G$
- for the weighted version G^C has a weight associated to each vertices $v \in V$ of value equal to its generalized degree *gendeg*(v) (see Definition 3.2.5)

Clique decomposition follows the idea of identifying and assigning the core part of the graph at first (also called sometimes the *backbone* which roughly represents the hardest part of the problem) and then extending it to the remaining part of the graph. Intuitively, this strategy is expected to be effective in order to solve FAP problems whenever the most connected part of the graph includes a considerable number of vertices. This will allow the partial assignments produced to be significantly reproduced, in terms of *building blocks*, in the optimal final assignment including all nodes in the network. However, clique decomposition is designed for a partitioning into only two components and different techniques need to be adopted to extend the method for more subsets. For example, in this thesis we adopt the criterion of iteratively finding the next largest clique after removing the vertices included in all of the cliques already found.

The algorithm find the *next clique* is simply that which removes from the graph the vertices of the cliques already found, and then applies the maximum clique procedure to this updated graph. This is formally stated in Algorithm 3.8.

Algorithm 3.6 Maximum clique unweighted recursive procedure (main)

```
Input:  $G^C(V, E)$, current size $s$, maximum size $max$, current depth $d$

1: $max = 0, d = 0, s = 0$
2: call MaximumCliqueUnweighted ($V, s, d, max$)
```

MaximumCliqueUnweighted ([27])

```
Input:  $U \subseteq V$, Integer $s, d, max$
Output: clique $C \subseteq G^C$
1: if $|U| = 0$ then
2:    if $s > max$ then
3:       $max \leftarrow s$
4:       Update current best clique ($C$) and store it
5:       $found := true$
6:    end if
7:    $d = d - 1$
8:    return $C$
9: end if
10: while $U \neq \emptyset$ do
11:    if $s + |U| \leq max$ then
12:       return $C$
13:    end if
14:    $i := min\{j \mid v_j \in U\}$
15:    $U := U \setminus \{v_i\}$
16:    $d = d - 1$
17:    call MaximumCliqueUnweighted ($U \cap N(v_i), s + 1$)
18: end while
19: return $C$
```

Algorithm 3.7 Maximum clique weighted recursive procedure (main)

```
Input:   $G^C(V, E)$, current size $s$, maximum size $max$, current depth $d$

1: $max = 0, d = 0, s = 0$
2: call MaximumCliqueWeighted ($V, s, d, max$)
```

MaximumCliqueWeighted ([96])

```
Input:   $U \subseteq V(G^C)$, Integer $s, d, max$
Output:  clique $C \subseteq G^C$
1: if $|U| = 0$ then
2:   if $s > max$ then
3:     $max \leftarrow s$
4:     Update current best clique ($C$) and store it
5:     $found := true$
6:   end if
7:   $d = d - 1$
8:   return $C$
9: end if
10: while $U \neq \emptyset$ do
11:   if $\sum_{k=1}^{d-1} w_{ki} + \sum_{i=1}^{m} w_{di} \leq \sum_{j \in C} w_j$ then
12:     return $C$
13:   end if
14:   $i := min\{j \mid v_j \in U\}$
15:   $U := U \setminus \{v_i\}$
16:   $d = d - 1$
17:   call MaximumCliqueWeighted ($U \cap N(v_i), s + 1$)
18: end while
19: return $C$
```

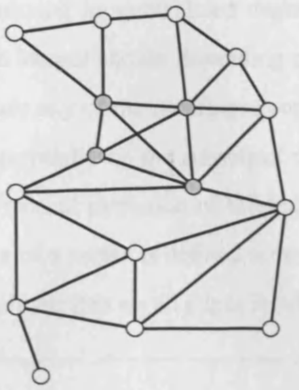


Figure 3.5: Example of clique decomposition into two subsets for a graph with sixteen vertices

Algorithm 3.8 Next clique procedure

```
Input:   Graph $G^C(V,E)$,  Integer $index$, number of subsets $n$
Output: $C \subseteq G^C$
 1: if $index \geq n$ then
 2:     $C \leftarrow G^C$
 3:     return $C$
 4: end if
 5: index=0
 6: $C \leftarrow$ call  Maximum clique ($G^C, index$)
 7: $G^C \leftarrow G^C \setminus G[C]$
 8: call  Next clique ($G^C, index + 1, n$)
 9: $C \leftarrow G^C$
10: return $C$
```

Figure 3.5 shows an example of a partition into two subsets produced by the (unweighted) clique decomposition for a graph with sixteen vertices.

3.2.5 Generalized degree

The *generalized degree decomposition* is based on the same criterion of solving the subgraph which roughly represents the hardest part of the problem first. The difference is that we here identify the core of the problem as the first N transmitters in a decreasing ordering by generalized degree of all the transmitters in the network, where N is an integer chosen depending on the desired size of the subset (alternatively we can use any of the orderings proposed in [54]). Note that, since Algorithm 3.1 acts sequentially on the subsets of the decomposition, this method can then be seen as a natural extension of the sequential assignment algorithms. The generalized-degree of a vertex is defined according to the definition proposed in [54] as 'the sum of all weights on all edges incident with a transmitter'.

Definition 3.4 *Given an unordered weighted graph G and a vertex $v \in V$ we defined the generalized degree of v as:*

$$gendeg(v) = \sum_{uv \in E} c_{uv}$$

The procedure can be extended easily to an arbitrarily given number and size of subsets and it has been successfully used in [25] to solve some MS-FAP and simple FS-FAP instances by applying an order-based genetic algorithm.

To summarise, the generalized degree procedure starts by ordering the whole set of transmitters before the division according the ordering which, among those proposed in [54], produces the best cost after a greedy sequential assignment. Subsequently, the chosen ordering is divided in a number of suborderings, which are obtained according to the following criteria:

- Divide the initial set of N transmitters (vertices) proportionally to the number $\frac{N}{M}$ of transmitters contained into each subset, where M is the number of subsets considered.

- Include in each subset the transmitters (vertices) which produce almost equal values of the corresponding sums under the graph generalized-degree versus transmitters

The latter algorithm, whose detailed procedure is given in Algorithm 3.9, produced the best results on the benchmark tested in [25]. An example of the application of the latter partitioning on a FAP benchmark is shown in Figure 3.6. Firstly, the diagram generalized-degree against transmitters is plotted (on the x axes transmitted are ordered by decreasing generalized degree). Than a partitioning is produced such that the values of the shaded areas differ only by a small given tolerance. Each of the areas represents the value of the sums S_j in Algorithm 3.9.

Algorithm 3.9 Generalized-degree Decomposition

```
Input:   G(V,E), Tolerance $t$ between the sums $S_j$
Output:  Partition { $V_1, V_2, \ldots, V_{nSubs}$ } of V into $nSubs$ sets $V_j$ of size $size_j$

1: Partition $V$ into $nSubs$ sets such as $S_h - S_k \leq t \, \forall \, h, k$ with the generic sum value for
   the $j^{th}$ subset given by:
       $S_j = \sum_{l=j}^{j+size_j} gendeg(v_l) \qquad j \in \{1, nSubs\}$
```

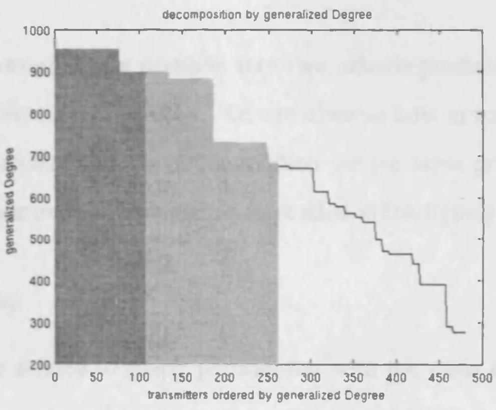


Figure 3.6: Partitioning in three subsets by generalized degree decomposition

This decomposition algorithm has the considerable advantage of its ease of implementation and a runtime which is comparable with the straightforward geographical and random decompositions mentioned above. However, as well as the other basic decomposition criteria, it does not include any information about the cost of the cut of the final partitioning produced and, as a consequence, the amount

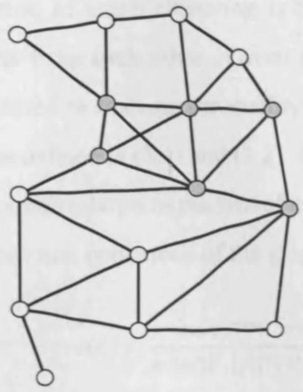


Figure 3.7: Example of generalized-degree decomposition into two subsets for a graph with sixteen vertices

of the inter-edges constraints that will be ignored by the returned solution. This may affect the quality of the final assignment produced by the meta-heuristic and could limit the use of this partitioning method for the simplest formulations of the FAP, such as the MS-FAP.

Figure 3.7 shows an example of a partition into two subsets produced by Algorithm 3.9 for a graph with sixteen vertices. We can observe how in comparison with the decomposition produced by the clique method for the same graph more vertices are added to the first subset (represented as shaded in the figure).

3.2.6 Graph clustering

Cluster analysis is closely related to graph partitioning with the main difference being that in the latter the required partition sizes can be specified in advance, whereas in the former the partition sizes freely varies. However, the effectiveness of a cluster strictly depends on some parameters, which are usually closely related to the number of clusters produced. As a consequence, clustering is quite often used only as a preprocessing step for the more clearly defined graph partitioning problem.

The natural definition of graph clustering is the separation of sparsely connected dense subgraphs from each other. Given a cut $c(G) = \{ V_1, V \backslash V_1 \}$ two indexes are usually defined to evaluate the quality: the *expansion* e and the *conductance* ϕ of the cut as defined in (3.1) and (3.2). If a cut has a small conductance it means that its size is small relative to the size of the smaller component it creates, so such a cut can be seen as a bottleneck of the graph.

$$e(c) = \frac{\sum_{uv \in E_1^{inter}} c_{uv}}{min(|V_1|, |V \backslash V_1|)} \tag{3.1}$$

$$\phi(c) = \frac{\sum_{uv \in E_1^{inter}} c_{uv}}{min(\sum_{uv \in (E_1^{inter} \cup E_1^{intra})} c_{uv}, \sum_{uv \in (E \backslash E_1^{intra})} c_{uv})} \tag{3.2}$$

We then extend the definition of the conductance to the whole graph G and a given partition of it.

Definition 3.5 *Given a partition into n subsets $C(G) = \{ V_1, V_2, \ldots, V_n \}$ we identify the cluster represented by the subset V_j with the induced subgraph of G:$G[V_j] := (V_j, E_j^{intra})$. The* conductance *$\phi(G)$ of a graph G is the minimum conductance value over all the possible cuts $c(G)$ of G. The* intra-cluster conductance *$\alpha(C)$ of $C(G)$ is the minimum conductance value over all induced subgraphs $G[V_j]$. The* inter-cluster conductance *$\delta(C)$ is one minus the maximum conductance value over all induced cuts $c_j = (V_j, V \backslash V_j)$:*

$$\alpha(C) = min(\phi(G[V_j])) \text{ and } \delta(C) = 1 - max(\phi(c_j))$$

The larger the *intra-cluster conductance* of a clustering, the higher the quality, since small intra-cluster conductance means that there is at least one of the clusters V_j containing a bottleneck which can be further decomposed into two subsets. A clustering with small inter-cluster conductance is also a low-quality one since there is at least a cluster with strong external connections. Optimising the indexes above is generally NP-hard as well as calculating the conductance of a graph, see [16].

In this thesis we have used the well known *Markov clustering* algorithm which is based on the intuition that dense regions in sparse graphs should correspond with regions in which the number of k-length paths is relatively large for small values of $k \in |V|$. As a consequence 'a random walk that visits a dense cluster will be unlikely to leave the cluster until many of its vertices have been visited' [120]. Pseudocode of the Markov clustering algorithm and a more detailed description of it can be found in [120].

It is important to mention that, although this is one of the most commonly used algorithms for graph clustering, the quality and number of clusters produced depends strictly on the expansion and inflation parameters. Furthermore, the convergence of the method is not always guaranteed. Finally, the algorithm is not able to produce a specified number of subsets. As a consequence, given the partition in a general number of k subsets produced by the Markov algorithm we obtain the desired decomposition into n subsets by further applying to the clustering avandongen (which will be described in the next subsection). Note that the size of the resulting subsets represented by the clusters is then generally unbalanced.

Figure 3.8 shows an example of a partition produced by the Markov clustering algorithm for a graph with twenty vertices reproduced from [120]. Here the ideal decomposition is that into four subsets indicated in the figure. To produce a partition into two subsets only a further graph partitioning procedure needs to be applied. This leads to the final decomposition into two subsets represented in the figure by shaded and non shaded vertices respectively.

3.2.7 Graph partitioning

The *Graph Partitioning Problem* is close to the ideal graph clustering described above but easily formulated and more often used in the literature. In a simple unordered graph this problem is defined as dividing the vertices into disjoint subsets such that the number of edges whose endpoints are in different subsets is mini-

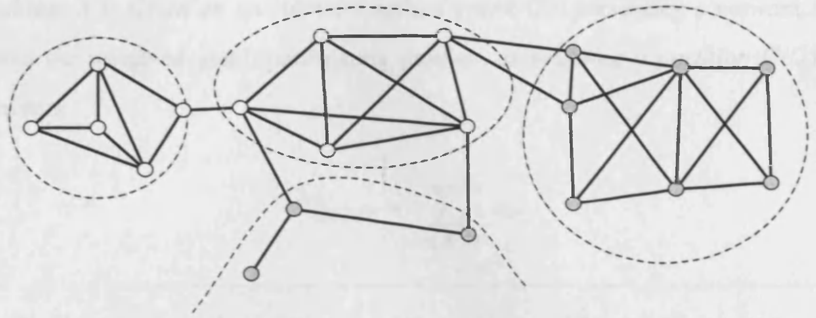


Figure 3.8: Example of Markov clustering decomposition into two subsets for a graph with twenty vertices

mized.

Definition 3.6 *We define a partition $C(G)$ of an unorderered graph $G(V, E)$ as a collection of n disjoint subsets:*

$$\{V_1, V_2, \ldots, V_n\} : \bigcap_{j=1}^{n} V_j = 0 \wedge \bigcup_{j=1}^{n} V_j = V.$$

We then represent the set of the external connections between distinct subsets as: $E^{inter} = \bigcup_{j=1}^{n} E_j^{inter}$.

Problem 3.2 *The* Graph Partitioning Problem *consists of minimizing the cardinality* $\mid E^{inter} \mid$.

If the graph is weighted, as is the interference graph, we reformulate the cost object of minimization as the sum of the weights of the external connections between different subsets.

Problem 3.3 *Given an unordered weighted graph G representing a network, we define the* weighted graph partitioning problem *as selecting a partition C(G) to minimize:*

$$C^1_{GPFAP} = \sum_{uv \in E^{inter}} c_{uv}$$

The *balanced graph partitioning problem* adds the additional constraint that the difference between the cardinality of different subsets must be as small as possible, i.e. either one or zero. Note that if we consider only two subsets this is known as the *bi-partitioning* problem which, when defined as unbalanced gives the *minimum-cut* formulated in Problem 3.1.

It is worth noting that the sum of inter-connections between different subsets in a given partition is not the only important measure of its effectiveness when subsequently used to solve FAP instances. Other factors can be equally important, such as the density of the intra-edges in the first (or first group) of the subsets. In fact, since during the first loop our procedure ignores all the constraints with the subsets yet to be assigned, whenever the first subset includes only a small number of vertices the meta-heuristic will rapidly find the optimal partial solution represented by an interference free assignment, thus limiting the effectiveness of the heuristic search itself. For this reason we have considered further formulations of the cost in order to lead to partitions for which the first subset aims to contain the most 'difficult' to assign transmitters .

This is also the idea used by the clique and generalized degree decompositions already introduced. As a consequence, we reformulate the graph partitioning problem in order to be more suitable for the FAP. We defined a new objective to be used in addition to the balanced constraint as follows (a similar cost function has been used in [26, 104]).

Problem 3.4 *Given an unordered weighted graph G representing a network, we* *define the* balanced Graph Partitioning Problem for FAP (GPFAP) *as selecting a partition $C(G)$ to minimize:*

$$C^2_{GPFAP} = \sum_{uv\in E} c_{uv} - \sum_{uv\in E_1^{intra}} c_{uv} + \sum_{uv\in E^{inter}} c_{uv}$$

such that the difference between the cardinality of different subsets is as small as possible, i.e $||V_h| - |V_k|| \leq 1\ \forall h, k$.

The cost function balances the sum of the weights of the external connections E_j^{inter} in each of the subsets with a term which maximizes the internal connections in the first subset only.

The objective in Problem 3.4 can be normalized to the total sum of the edge weights for computational convenience:

$$C^2_{GPFAP} := \frac{C^2_{GPFAP}}{\sum_{uv\in E(G)} c_{uv}}$$

We have also formulated an unbalanced GPFAP which removes the balanced constraint. For this problem better performance is obtained if we aim to minimize the external connections between pairs of different subsets while maximizing the internal connections in each of the subsets. In fact, this encourages the size of each subset to be relatively balanced in order to prevent the tendency of including a very high percentage of transmitters in one subset only. For this purpose, we define a different objective based on the conductance indexes introduced for the graph clustering. In fact, given a partition $C(G)$ we can easily compute the conductance $\phi(c_i)$, see Definition 3.5, over all its induced cuts. Since we aim to minimize this value for all the cuts in a cluster we define the following objective:

Problem 3.5 *Given an unordered weighted graph G representing a network, we define the* unbalanced Graph Partitioning Problem for FAP (GPFAP) *as selecting a* *partition $C(G)$ to minimize:*

$$C^3_{GPFAP} = \sum_{i=1}^{n} \phi(c_i)$$

where:

$$\phi(c_i) = \frac{\sum_{uv \in E_i^{inter}} c_{uv}}{min(\sum_{uv \in (E_i^{inter} \cup E_i^{intra})} c_{uv}, \sum_{uv \in (E \setminus E_i^{intra})} c_{uv})}$$

To avoid trivial solutions for the first partial assignment the subsets are reordered in a decreasing order of size.

Finally, although both objectives in Problems 3.4 and 3.5 can be used for both the balanced and unbalanced types of GPFAP, in the rest of this thesis we will only present the results produced by those best performing on some preliminary runs (see [26]).

Figure 3.9 shows an example of a partition produced by the graph partitioning algorithm for a decomposition into two subsets for graph with twentysix vertices using each of the costs proposed. The first cost 3.3 only founds the partition into the two disconnected component. Note that the same decomposition would be found by the minimum-cut algorithm described in Section 3.2.3. Cost 3.4 produces the balanced partition into the two subsets of size of thirteen vertices each. Finally the cost 3.5 returns, after discarding the trivial solution constituted by the disconnected components, an unbalanced decomposition with only seven vertices in the subset of minimum size. Note that in this case the disconnected components are included in the biggest subset instead.

A memetic GA for GPFAP

To solve the GPFAP in all the formulations introduced in the previous subsection we have implemented a memetic GA. The aim is to obtain near optimal solutions in a reasonably short time rather than pursue the absolute optimal, since the partitioning only constitutes the preprocessing step of the the subsequent procedure

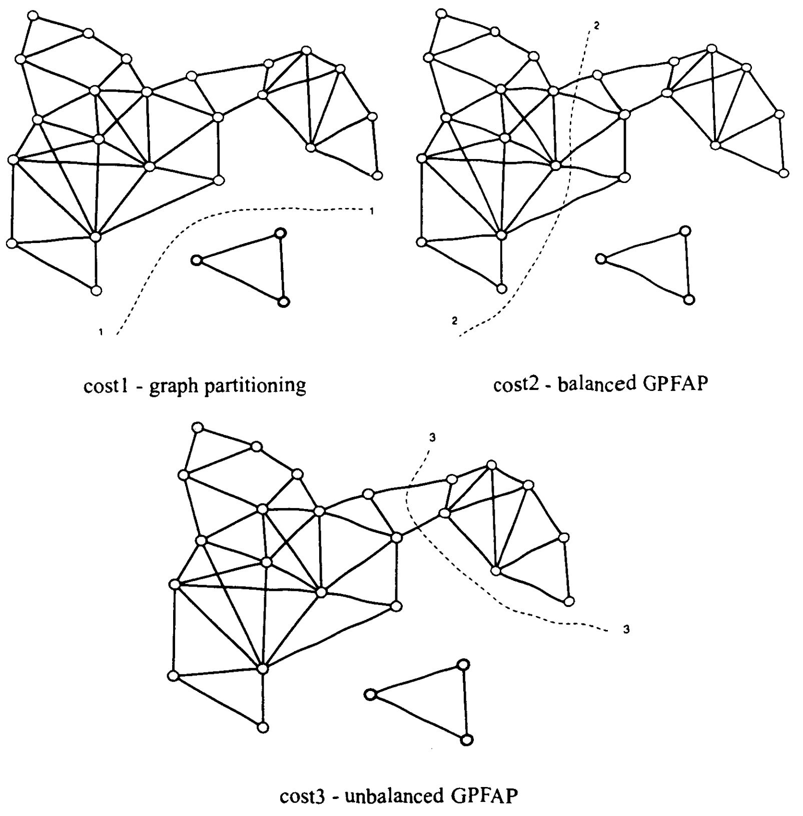


Figure 3.9: Example of graph partitioning into two subsets for a graph with twentysix vertices

which solves the FAP. We have implemented a standard version of SEAMO, the steady-state GA originally proposed in [119], which has been shown to be very effective with the order-based representation also used in this context [23].

With this representation individuals are represented by a permutation of the transmitter set and the fitness evaluation procedure consists of decomposing a given ordering into M subsets, whose size is also chosen at random, and then computing the desired objectives defined above. The genetic operators used were those suggested in [23] to obtain the best performance, that is *cycle crossover* and *order-based mutation*. The cycle crossover operator is used with chromosome representations which consist of permutation of integers. It identifies a number of so-called cycles between two parent chromosomes.

For a more detailed description of the method and examples see [118] and Appendix C.2.2. Order-based mutation simply consist in swapping two elements of the permutation representing a single chromosome.

The choice of parents for mating is conducted by the so called *roulette wheel selection*. Here the probability of selection for the chromosomes in the population is computed proportionally to their fitness values. In other words, if F_i represents the fitness of each individual in the population, its probability of being selected is

$$p_i = \frac{F_i}{\sum_{j=1}^{popSize} F_J}$$

where *popSize* is the number of individuals in the population.

To prevent the problem of *premature convergence*, which means that the population converges too early resulting in being suboptimal, we have implemented a *fitness sharing* procedure. In a multimodal domain in which more then one peaks (local optima) are present standard GAs will eventually converge only to one of those peaks This phenomenon is also known as *genetic drift* [37].

A multimodal domain can be thought as as formed by niches as an analogy fro nature in which the number of organism contained in a niche is determined by both the fertility of the niche itself and the efficiency of each organism to exploit such fertility. If tee are too many organisms within a single niche there will be no provisions for everybody and the least efficient organisms will die. Conversely, it there are only few organisms in a fertile niche these will quickly reproduce to fully exploit the niche's resources [85]

Fitness sharing is one of the techniques proposed to force a genetic algorithm to preserve diversity in a population throughout its search in order to avoid convergence to a single peak. This method was originally proposed by Goldberg and

Richardson [48] and it consists of a segmentation of individuals of similar fitness in specific groups. (*niches*) Here the main idea is that of penalizing the fitness of solutions which are too similar to each other.

In the following equations F_i represents the real fitness of an individual (here represented by the costs O_{GPFAP}) and m_i is the so-called niche count calculated by summing a sharing function $sh(d_{ij})$ over all individuals j in the population. This is a function of the distance d_{ij} between two individuals i and j computed according to some metrics to represent distances between permutations. In our implementation this distance is calculated in the objective space (also called 'phenotype level') and it is simply equal to the euclidean distance between the fitness values F_i and F_j. In more expensive implementations the distance d_{ij} can be defined directly between the chromosomes, here represented by permutation of integers. This is also known as 'genotype level'. More details about metrics for distances between permutations can be found in [103]. If the distance d_{ij} is within a fixed radius σ_{sh} the sharing function $sh(d_{ij})$ returns a value between 0 and 1 which decreases with d_{ij} (i.e. with increasing similarity). Otherwise it returns 0.

$$\begin{aligned} F_{sh,i} &= \frac{F_i}{m_i} \ \textit{if maximization problem} \\ F_{sh,i} &= F_i \cdot m_i \ \textit{if minimization problem} \end{aligned} \tag{3.3}$$

$$m_i = \sum_{j=1}^{n} sh(d_{ij}) \tag{3.4}$$

$$d_{ij} = | F_i - F_j | \tag{3.5}$$

$$sh(d_{ij}) = \begin{cases} 1 - (\frac{d_{ij}}{\sigma_{sh}})^{\alpha} & \textit{if } d_{ij} < \sigma_{sh} \\ 0 & \textit{otherwise} \end{cases} \tag{3.6}$$

The parameter α is conventionally set to 1 while the suitable value for σ_{sh} can be calculated given the expected number of peaks in the domain and the hyper-volume of the entire domain space as proposed by Deb and Goldberg in [35]. Given a set of points in the domain a definition of hyper-volume is given as 'the Lebesque measure of the set of all points that are Pareto dominated by at least one of the given points', see [73] for details.

However, σ_{sh} is usually conservatively estimated, depending on the particular test problem considered, as the minimum niche radius of any optimum within the domain (see [48]). In this thesis we assumed a value of $\sigma_{sh} \simeq \frac{F_{min}}{100}$, as suggested in [25].

Although it is not necessary to solve the GPFAP exactly, test runs have shown that better approximations in the decomposition lead to better FAP solutions. Consequently, to speed up the process for MI-FAP instances and to improve solution quality also, the GA has been hybridised with a local search procedure after each individual evaluation, in order to search for local optimality. We have implemented a standard SA (as described in [60]), in which each move consists, as well as for the mutation operator for the GA, in a single transmitter swap between two different subsets. Pseudocode for the memetic GA is given in Algorithm 3.10.

Finally, it is important to specify that for cellular problems (as the COST259 benchmarks of the MI-FAP) the GPFAP procedure has been implemented in terms of single cells rather than single transmitters, that is the vertices of the equivalent graph G^D are the network cells instead of the transmitters vertices. This automatically preserves the co-cell constraints for each of the subsets.

Algorithm 3.10 Memetic GA for the Graph Partitioning problem.

```
Input:   population size popSize, number of subsets nSubs,
         number of generations $G$, number of iterations $I$, cost $C_{GPFAP}$
Output:  Partition { $V_1$, $V_2, \ldots, V_M$ } of V

 1: Generate a population of popSize individuals as random permutations of the whole set
    of transmitters\cells representing a chromosome
 2: Decompose each ordering into nSubs subsets with either fixed or randomly variable size
 3: Evaluate the fitness $C_{GPFAP}$ for each individual
 4: Store the bestSoFar fitness value
 5: while stopping condition not satisfied do
 6:   while next individual in the population do
 7:     This individual becomes the first parent
 8:     Select a second parent by applying roulette wheel selection
 9:     Apply cycle crossover to produce offspring
10:     Apply order-based mutation to offspring
11:     Evaluate fitness of offspring
12:     Apply SA for $I$ iterations to improve local optimality
13:     Add fitness sharing to each of the offspring
14:     if offspring better than either parent then
15:       Replace the weakest parent
16:     else
17:       Randomly select another individual in the population and replace it if it is weaker
18:     end if
19:     Update bestSoFar
20:   end while
21: end while
22: Return the decomposition representing the bestSoFar individual
```

Chapter 4

Evaluation and benchmarks

In the first section this chapter discusses how a FAP solution produced by the decomposed assignment approach will be evaluated. This technique will be compared to other solutions obtained by a heuristic applied in the traditional way, in which the problem is solved as a whole data set. The aim is to determine which technique is more effective in practical application. As such, both the runtime and the optimality of assignments produced must be considered simultaneously.

Subsequently, the chapter outlines the benchmarks to which the decomposed assigned technique will be applied. These are divided into two groups for the MS/FS-FAP and for the MI-FAP respectively.

4.1 Evaluation of a decomposition

4.1.1 Quality

A first basic evaluation strategy is that which compares different decomposition techniques exclusively in terms of optimality of the results produced. This criterion compares the final costs produced by the decomposed with the whole approach. To avoid significant variations in runtime each of the FAP instances is solved over the same number of evaluations, that is the number of different configurations explored by the heuristic. Intuitively, this represent a fair criterion for comparison between the two approaches.

According to Algorithm 3.1 and 3.2, the decomposed procedure can loop through the subsets a number of times, and for each of them evaluates a defined number of assignments.

Definition 4.1 *Let $eval_{i,j}$ the number of evaluations corresponding to the i^{th} subset and the j^{th} loop. Let $eval_{tot}$ the number of evaluations corresponding to the solution obtained with the non-decomposed approach. In order to have the same number of* *evaluations we have to respect the following condition:*

$$\sum_{j=1}^{nLoops} \sum_{i=1}^{n} eval_{i,j} = eval_{tot}$$

Note that whenever the subsets are solved independently using Algorithm 3.2 a single evaluation of the whole set of transmitters (calculated with respect to the specific representation used for the frequency assignments) needs to be added at the end of the decomposition procedure, i.e. the condition to respect becomes:

$$\sum_{j=1}^{nLoops} \sum_{i=1}^{n} eval_{i,j} + 1 = eval_{tot}$$

Hence, given the number of subsets n is fixed in a given experiments, we can then reach a desired number of evaluations by changing either of the two inversely proportional variable *numL* and $eval_i$. For instance, if we want to loop through the subsets only once we can either reduce the number of evaluations proportionally to the number of subsets ($eval_{i,0} = \frac{eval_{tot}}{n}$) or to the number of vertices contained in each subset ($eval_{i,0} = \frac{eval_{tot} * |V_i|}{|V|}$).

A different approach consists of aiming to improve the results by looping through the subsets more than once, thus reducing the number of evaluations computed for each subset proportionally to the number of loops. Finally, the equality in Definition 4.1.1 refers to a single run of the heuristics. Therefore on a given data set and for a given decomposition method multiple runs of the same heuristics need to be conducted and averaged.

4.1.2 Computational complexity

A second straightforward evaluation criterion consists in analysing only the computational complexity of the decomposition procedure, and, as a consequence, estimating its runtime. Of course a faster runtime does not necessary mean anything about the validity of the approach if not accompanied by an evaluation of its quality. However, in some situations we aim to find only approximated solutions rather than the optimal. This is the case of very large and complex problems in which good approximations of the solutions are more than acceptable, since it is impossible to solve the problem as a whole in a reasonable runtime.

Although we are expecting some advantages with the decomposed approach, a correct detailed analysis which considers all the factors involved in this meta-heuristic technique is far too complex. To conduct a complete evaluation of a given assignment with the global approach we need to compute all the edges of the interference graph $G(E, V)$. However, with the decomposition approach when we are computing an assignment of the i^{th} subset we deduce from Algorithms 3.1 and 3.2 that we have to compute only its intra-edges E_j^{intra} and the inter-edges E_j^{inter} for that subset, thus ignoring all the remaining edges of G.

Note that the computations required are smaller for the 'independent' approach in 3.2 and the first loop of the 'sequential' procedure in 3.1, since we need to evaluate only partial assignments in which either the whole set of the inter-connections E_j^{inter} or a part of it is ignored.

However, in some circumstances we do not need *complete* evaluations (that is to compute the edges of the neighbours of all the vertices included into the current subset) but only an *update* of the current assignment. This is, for instance, the case improvement methods in general and in particular our implementation of SA in which a single move consists of one change in the frequency of a selected vertex, thus only requiring the computation of the edges in its neighbourhood. In this cases, benefits in computational time will be given only within the first loop through the subsets in which a part of the inter-edges between pairs of distinct subsets is ignored.

Furthermore, although the evaluation of an assignment is the most computationally expensive part of the heuristic procedure, this is not the only aspect which can be affected by the decomposed assignment approach. For example, for the

GA we will have the application of the genetic operators (crossover and mutation) as well as other procedures, as for example sorting algorithms. These generally needs a lower number of basic operations since they are applied to chromosomes of shorter lengths than the entire set of vertices of the graph.

On the contrary with local search heuristics such as SA the benefits provided by these additional features are more limited. For example, with SA they only come from copying vectors of smaller sizes (see Appendix C.2.1) in order to record the best assignment currently produced. As a consequence for this category of heuristics becomes crucial to analyse the trade-off between quality of a solution and run time of execution.

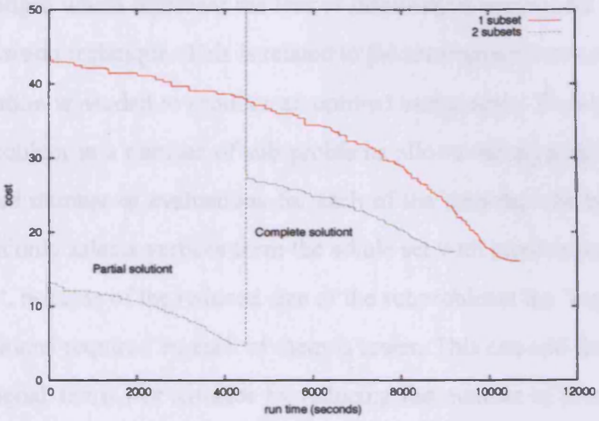
4.1.3 Trade-off between quality and runtime

Neither of the two methods introduced above seems able to define an absolute criterion for evaluating the effectiveness of a given decomposition strategy. In fact, quality evaluation is unlikely to be successful with enough generality over the set of benchmarks whereas a possible evaluation based only on the computational complexity does not consider any issues about the optimality of the solution produced. In order to fulfill both these requirements this section will analyze other possible criteria which take into account the trade-off between quality and runtime of an approximated solution produced by the decomposed assignment approach applied to the FAP.

A first immediate way of considering this is by plotting in the same diagram the curves of quality (in terms of cost produced) versus runtime for different runs of the same heuristic using either the decomposed or the non-decomposed approach. For instance, we can plot on the same graph a single run of a given meta-heuristic applied to the problem solved as a whole, and compare it with another run of the same algorithm obtained with Algorithm 3.1. We can then evaluate at a fixed time which of the solutions produces the better cost.

An example of this is shown in Figure 4.1 which shows two runs of SA for the MI-FAP Siemens2 benchmark obtained by the non-decomposed approach and with a decomposition into two subsets. Although at the end of their runs the two approaches produce a similar cost, we can appreciate that the cost-time curve produced by the decomposed technique is always below the other for any fixed time, thus making this approach preferable for this particular test run. However, only the assignments in the last of the subsets are valid solutions representing complete assignment of the whole set of transmitters in the network, whereas the other subsets only produces partial assignments in which some vertices of the graph are still unassigned.

Figure 4.1: Cost-Time plot for Siemens2 solved by SA with balanced graph partitioning decomposition into two subsets after $2,000,000 * |V|$ evaluations



It is important to note that, according to Definition 4.1 each of the runs of the given test problem corresponds to a same given total number of configurations explored ($2,000,000 * |V|$ evaluations in the example), thus they are expected to produce a similar run time.

Section 4.1.2 has shown that, whereas evolutionary algorithms take more advantages from a reduced computational complexity, for local search techniques this mainly derives from ignoring the inter edges between subsets. Thus if the number of these tends to zero the two approaches (global and decomposition) are expected to produce very close run times whereas many inter-edges give more advantages to the decomposition approach. On the contrary, ignoring a high number of inter-edges may degrade the performance of the decomposition approach. In fact, the quality of its solutions can only be greater or equal to that of the whole approach, i.e. it is equal when there are no inter-edges. However, the latter statement is valid only if we assume infinite run times for both approaches.

When we limit the number the number of total evaluations to a fixed value instead, the decomposition approach is expected to produce some advantages in quality terms too. In fact, we have seen in Section 2.2.2 that the relation between the 'expected' time to produce an optimal solution and the size of the problem follows a non linear (exponential) behaviour. This implies a better performance of the decomposition approach the more the fixed time is shorter than the 'expected' one provided the finding of a 'good' decomposition, i.e. one not ignoring a high number of inter-edges which represent the loss of information consequent to the use of the decomposition technique. This is related to the assumption that a specific number of evaluation is needed to produce an optimal assignment. Thus decomposing the whole problem in a number of sub-problems allows the algorithm to conduct a certain fixed number of evaluations for each of the subsets, whereas the whole approach can only selects vertices form the whole set with random probability.

Note that, because of the reduced size of the subproblems the 'expected' number of evaluations required by each of them is lower. This can add further benefits in computational terms, for instance by reducing the number of time we have to record the best current solution, see Section 4.1.2. Moreover, we could even stop the algorithm after a fixed number of iterations without any improvement in the best current solution. Therefore the decomposition approach may require less than the fixed number of evaluation used for the global approach in order to produce a nearly optimal solution, i.e the equation in Definition 4.1 would become the inequality 4.1, with consequent advantages in runtimes.

$$\sum_{j=1}^{nLoops} \sum_{i=1}^{n} eval_{i,j} \leq eval_{tot} \tag{4.1}$$

Note that the advantages outlined above are lost if we assume that a specific benchmark could be solved by an exact procedure. In fact, with exact algorithms the solution produced by the decomposition approach cannot be better than that of the global approach, i.e. it is equal when all subsets are disconnected components with no inter-edges. Note that this is also a case in which the two approaches are expected to produce exactly the same run time since the number of constraints explored by the two procedures in identical.

Advantages in run time can be obtained from the decomposition when some inter-edges representing constraints between distinct subsets are present, although this is expected to produce only approximations of the exact solution, whose quality depends on the amount of constraints ignored. Note that, this is the procedure used in real applications by the operators when they split the network into smaller areas (see for example the case of London), which assumes a relatively low number of inter-edge constraints. However, to the author's knowledge the decomposition procedure used is only based on geographical information (see also Section 3.2.2).

Nevertheless, despite the disadvantages described above, decomposition techniques have been succesfully used in combination with exact methods too to reduce the size of MS-FAP and FS\MI-FAP instances [3, 81], and to produce lower bounds [82].

To summarize we can then record the pair values (cost, runtime) in a diagram for both the solution obtained by the whole approach and that produced by the decomposition into two subsets. We can then repeat the same procedure, for both the decomposed and the whole approach, for other runs corresponding to different numbers of total evaluations, thus obtaining other pair values. Furthermore, we can repeat the decomposed assignment procedure for an increasing number of subsets obtaining different points in the diagram.

Intuitively, we are expecting the best quality being still produced either by the non-decomposed approach, or by the solutions obtained by a decomposition into a small number of subsets. As the number of subsets increase the approximation produced are likely to worsen rapidly (that is more than proportionally) since it becomes increasingly harder to limit the number of the inter-connections E_j^{inter} between different subsets. Note that if n subsets are used the decomposed assignment approach coincides with a sequential algorithm described in Section 2.2.1.

Hypothetically, we will aim to obtain in the diagram the non linear curve presented in Figure 4.2. Note that the point of minimum value will represent the ideal number of subsets for that particular benchmark problem. However, the cost produced by the heuristic procedure increases considerably when a high number of subsets is used (in some cases even for a limited number of subsets, e.g. $\geq$ 3). As a consequence the resulting diagram will consist of a series of curves, each of them corresponding to a fixed number of evaluations, as represented in Figure 4.3 with the lines in blue. This behaviour would make more difficult the prediction of the most suitable number of subsets for a given data set. An example of this will be discussed in Section 7.5.

Figure 4.2: Trade-off of quality against runtime for a generic test problem decomposed into two and three subsets. Hypothetical curve. n number of subsets, n_e number of total evaluations

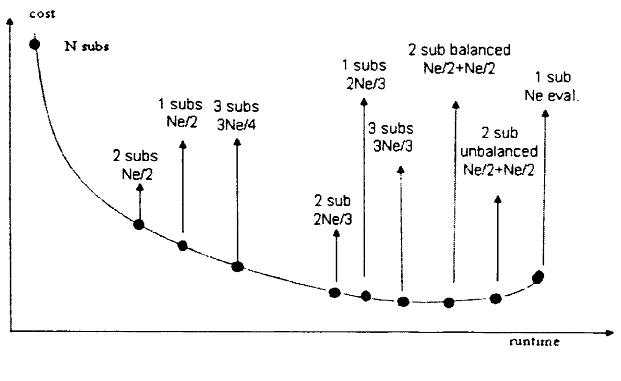
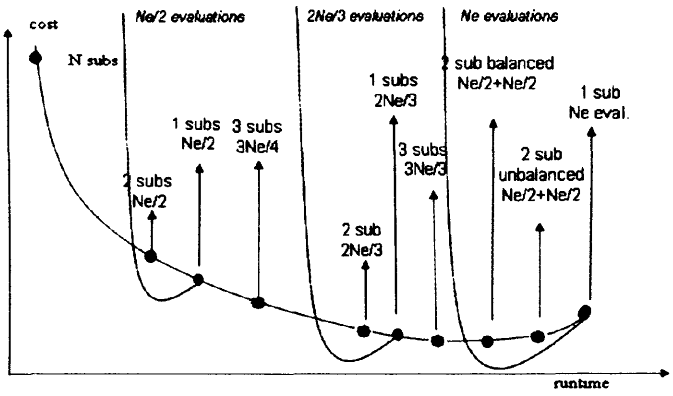


Figure 4.3: Trade-off of quality against runtime for a generic test problem decomposed into two and three subsets. Practical curve. n number of subsets, n_e number of total evaluations



4.2 Benchmark data sets

4.2.1 Benchmarks for MS-FAP/FS-FAP

The most commonly published benchmarks for MS-FAP are the cellular Philadelphia instances [10], which, although obsolete, are still currently used as a testbed for the MS-FAP. Other notable published data sets are a part of the military applications of the CALMA project [5] (also used for the FS-FAP). However, these will not be included in this thesis.

Philadelphia and cellular problems

For both the MS-FAP and the FS-FAP described later in this chapter, we refer to the same set of test problems, which primarily coincide with those used in [25]. All of the data sets outlined in the following adopt the simple *binary constraint* model formulated in Section 1.2. Firstly, two simple test problems (called P_1, P_2 in conformity with [119]) in the order of hundreds of transmitters have been taken for comparison from the data sets used in [118]. Transmitters are located in cells forming an hexagonal grid and those belonging to the same or adjacent cell are not allowed to use the same frequency. However, variants of this structure use different re-use distances (with distances expressed in 'units') $(d_0, d_1, ..., d_n)$. This means that transmitters in cells of distance d_0 (measured from cell centres), can use the same frequency, those within distance d_1 require a separation of one channel, etc. Due to the very high demand assigned to each cell, the resulting interference in the whole network will be primarily governed by the *co-cell* interference, that is, by the constraints between pairs of transmitters within the same cell. Figure 4.4 shows the grid model and the required cell demands for Philadelphia P_2, which has the following demand vector c_v and reuse distance d_v.

$$c_v = \{8, 25, 8, 8, 8, 15, 18, 52, 77, 28, 13, 15, 31, 15, 36, 57, 28, 8, 10, 13, 8\}$$

$$d_v = \{\sqrt{12}, \sqrt{3}, 1, 1, 1, 0\}$$

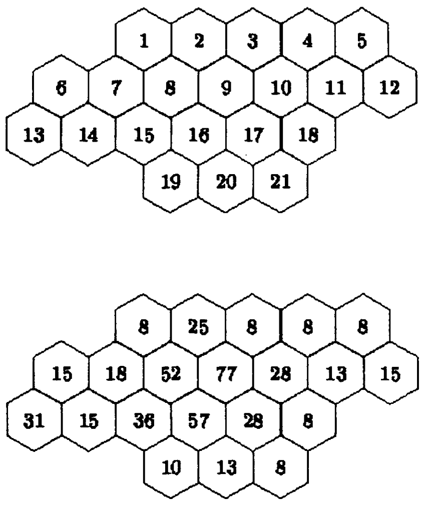


Figure 4.4: Hexagonal grid model (a) and cell demand (b) for Philadelphia P_2.

The first cell requires a demand of 8 transmitters, the second one of 25 and so on. Two transmitters assigned to the same frequency must be distant at least $\sqrt{12}$ units apart to avoid interference (where one unit is equal to one cell). If the transmitters are assigned to adjacent frequencies they must be distant at least $\sqrt{3}$ units, transmitters with three channels separation must have a distance of at least one unit and so on. Figure 4.5 shows the reuse distances for P_2 with the number inside the cells representing the minimum channel separation required from the central cell.

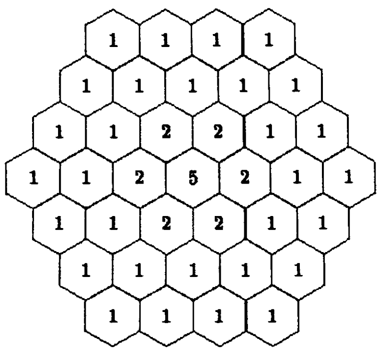


Figure 4.5: Reuse distance for Philadelphia P_2.

The smaller benchmark P_1 is a computer generated problem based on the same cellular principle. For more details see [118].

Cardiff University data sets

More realistic problems were created using a *benchmark generator* tool described in [9]. This software uses a probabilistic modelling to produce realistic transmitter locations over geographical areas and generates appropriate constraints. It places transmitters according to a probability distribution function obtained by building up single gaussian distributions centered in particular locations representing towns over a fixed region. Test problems C_1 and C_2 were generated by using a 'single-town' probability distribution whereas C_3 and C_4 by a distribution over 'two-towns', with the distance between the towns center greater in the last problem. Transmitter locations for data sets $C_1 - C_4$ are shown in Figure 4.6. Constraints are generated by defining a reuse distance vector requiring separations up to two frequencies ($d_v = (d_0, d_1, d_2)$).

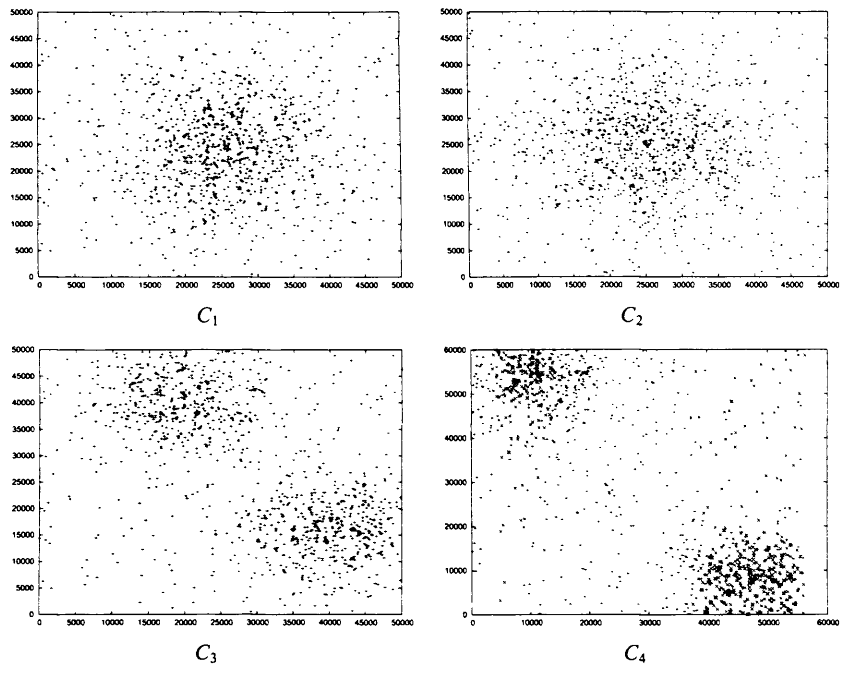


Figure 4.6: Transmitter locations for data sets C_1 – C_2 generated with a 'single-town' probability distribution

Other GSM

Besides these data sets, another group of benchmarks G_1, G_2, and G_3, which represent real anonymous GSM scenarios, were also provided by Cardiff University. However, no geographic information about transmitter locations is available. Test problems G_1 and G_2 refer to the same data set with identical number of transmitters but different constraint graph connectivity.

Random Graphs

We include a final group of data sets belonging to the category of random graphs. Due to their peculiar structure which often implies very high graph density, random graph do not represent the type of problem likely to occur in practice. In order to construct a random graph we need to define some auxiliary variables.

Definition 4.2 *We define $p_i \in [0, 1)$ $0 \leq i \leq nSubs$ $(p_0 = 0)$ to be the probability that the minimum required frequency separation between a pair of vertices u, v is $i + 1$. Clearly we require that:*

$$\sum_{i=0}^{nSubs} p_i \leq 1.$$

Let $q = 1 - \sum_{i=0}^{n} p_i$ denote the probability that there is no constraint between a pair of vertices. We then define a second set of variables $p_k' \in [0, 1)$ such that:

$$p_k' = \sum_{i=0}^{k} p_i$$

Algorithm 4.1 gives the pseudocode of the procedure used for the generation of random graph data sets. Note that for consistency with the other benchmarks the adjacent matrix of the graph remains symmetric.

Algorithm 4.1 Random graph

```
Input:   Int size, Vector $\{p_0, p_1, p_2\}$
Output:  Graph $G(V, E)$
1: for $i = 1$ to size do
2:    for $j = i + 1$ to size do
3:       Select a random value $rn \in [0, 1)$
4:       if $0 \leq rn < p_{nSubs}'$ then
5:          Choose $k \in \{0, 1, ..., nSubs - 1\}$ such that $p_k' \leq rn \leq p_{k+1}'$
6:          Add edge $ij$ with weight $k$
7:       else
8:          Add no edge
9:       end if
10:   end for
11: end for
```

Algorithm 4.1 has been used to generate two random instances with the same size of 500 transmitters. The first data set R_1 considers only co-channel interference distributed with a probability $p_0 = 0$, $p_1 = 0.75$, $q = 0.25$, whereas a second test problem R_2 takes into account one, two, and three channel separations with respective probability values expressed by the parameters $p_0 = 0$, $p_1 = p_2 = p_3 = 0.25$, $q = 0.25$.

Benchmark comparison

All of the test problems considered are detailed in Table 4.1. According to the convention used in [64], for each of them the size of the search space Sz is calculated with a hypothetical fixed spectrum of 1000 frequencies. A measure C_n of connectivity is expressed by the edge density of the corresponding constraint graph (a value of one represents full connectivity). Finally, the last column shows the spectrum available chosen for the FS-FAP experiments conducted for this thesis.

$$C_n = \frac{2|E|}{|V|(|V|-1)} \in ([0,1]$$

	No trans.	No const.	Sz	C_n	Spectrum
P_1	97	1214	10^{285}	0.26	30
P_2	481	97835	10^{1443}	0.847	30
C_1	1054	40200	10^{3162}	0.072	65
C_2	1068	39122	10^{3204}	0.068	65
C_3	1089	38869	10^{3267}	0.065	50
C_4	1112	34704	10^{3336}	0.056	50
G_1	1667	28455	10^{5001}	0.020	30
G_2	1667	338611	10^{5001}	0.250	30
G_3	1668	103268	10^{5004}	0.074	30
R_1	500	99831	10^{1500}	0.8002	150
R_2	500	99852	10^{1500}	0.8004	150

Table 4.1: Benchmark test instances for MS-FAP and FS-FAP

Note that two groups of benchmarks presents a particularly high density, i.e. the Philadelphia instances (due to their very high cell demand) and the random graphs.

Table 4.2: Number of transmitters per cell in the Tiny instance

cell	A_1	A_2	A_3	B_1	B_2	C_1	C_2
transNo	1	3	2	2	1	1	2

4.2.2 Benchmarks for MI-FAP

COST-259

The main group of test problems selected for this type of FAP consists of the COST259 instances introduced in Section 2.4. This project, widely available from [2], probably constitutes the most important set of benchmarks used in the last decade to study the FAP. These data sets are still currently used in practice and/or papers and were explicitly designed with the purpose of comparing, improving and developing new assignment methods.

Partners within COST259 have proposed their own benchmarks. Each of the resulting data sets presents the same structure briefly described in the following by considering the small test example called Tiny (see [2, 28, 40]). This small benchmark comprises three sites: site A has three sectors 1, 2, and 3, whereas sites B and C have only two sectors, 1 and 2. A cell is served by one sector of one site. The numbers of elementary transceivers ($TRXs$) installed per cell are given in Table 4.2.

We can interpret Tiny as a GSM900 network with radio frequency band $891.0-893.4MHz$ available for the downlink and band $936.0-938.4MHz$ available for the uplink. These result in thirteen different channels which have the Absolute Radio Frequency Channel Numbers (ARFCNs) of 711 – 723. We refer to these channels as the *frequency spectrum*.

Due to technical and regulatory restrictions, some channels in the spectrum may not be available for some of the cells, thus are called *locally blocked*. Local blockings can be specified for every cell. For each of the cells we number the carriers starting at zero, with the first carrier operating the *broadcast control chan-*

nel (BCCH) and the other possible carriers operating the *traffic channel*s (TCHs). BCCH is the control channel which contains specific parameters needed by a mobile in order to identify and access the network. Since no radio frequency hopping is considered here, carriers and transmitters follow a one to one relation.

The difference of the ARFCNs of two channels is a measure for their proximity and for each pair of transmitters there may be the constraint of a separation requirement in order to avoid strong interference. Separation requirements are not allowed to be violated in a channel allocation and constitute *hard constraints*. These can be further classified into *co-site* constraints (two channels separation), for transmitters belonging to the same site but different cells, and *co-cell* constraints (three channels separation), for transmitters belonging to the same cell, and further constraints due to *hand-over*. During hand-over an ongoing call is passed through adjacent cells and the communication switches from the channel used in between the current cell to one of the available channels used by the transmitters belonging to the receiving cell. Hand-over relations are represented as in Table 4.3, which indicates the cells that can be subjected to it. As a consequence some separation between channels in the two cells allowing hand-over is required. Typical values are expressed in terms of separations between BCCH and TCHs as shown in the example in Table 4.4. The BCCH and all TCHs in the passing-on cell have to be separated by at least 2 channels from the BCCH in the receiving cell. Whereas, the BCCH and all TCHs in the passing-on call have to be separated by only 1 channel from the TCHs in the receiving cell.

Between transmitters installed at different sites, only co-channel and adjacent-channel interference is relevant. This "acceptable" interference constitutes the *soft constraints*. As specified in [40], for computational convenience this interference is specified only for the down-link band, which usually occurs between transmitters in different sites.

Interference relations are usually not symmetric and expressed in terms of affected cell areas, which are further normalised between 0 and 1. There are sev-

Table 4.3: Hand-over relations for the Tiny instance

	A_1	A_2	A_3	B_1	B_2	C_1	C_2
A_1		●	●				
A_2	●		●	●			
A_3	●	●				●	●
B_1		●			●		●
B_2				●			●
C_1			●				●
C_2			●			●	

Table 4.4: Hand-over separation for the Tiny instance

	BCCH	*TCH*
BCCH	2	1
TCH	2	1

eral ways to rate interference including area-based and traffic-based ratings. The COST259 scenario refers to area-based as a measure of the acceptable interference. Table 4.5 specifies the interference values for the Tiny example which are specified in terms of pairs of cell rather than pairs of transmitters.

Soft and hard constraints are combined in order to produce the final interference model for the MI-FAP described in Problem 1.3. If a minimum separation of one is required by a hard constraint this actually excludes co-channel interference since the specific pair of transmitters cannot use the same channel. In the same way a separation of two or more channels excludes both co-channel and adjacent channel. As a consequence, the co-channel and adjacent channel values in Table 4.5

Table 4.5: Co-channel (left) and adjacent-channel (left) interference between cells in the Tiny instance

	A_1	A_2	A_3	B_1	B_2	C_1	C_2
A_1							
A_2				(0.30, 0.10)	(0.10, 0.02)		
A_3						(0.05, 0.00)	(0.20, 0.06)
B_1	(0.01, 0.00)	(0.25, 0.09)					(0.25, 0.08)
B_2							(0.15, 0.04)
C_1			(0.01, 0.00)				
C_2		(0.06, 0.01)	(0.12, 0.03)		(0.25, 0.08)		

will be substituted by a suitable large value c_e^{hard} where there is a hard constraint.

The COST259 instances use the sum of interference measures over all pairs of transmitters in the network as the global objective of minimization. Although this criterion can be considered inadequate, since it does not take into account in any form the local distribution of interference, it has nevertheless been proven as effective in practical terms. The calculation of the cell capacities, and thus the number of transmitters contained in each cell, depends on the expected traffic load.

Among the 32 planning scenarios available provided by different sources we have selected the following group of data sets available from [2]:

Bradford nt-*t*-eplus *(provided by E-Plus Mobilfunk GmbH)* A GSM 1800 network with 649 active sites and 1886 cells. The parameter t stands for five different traffic loads. The basic traffic load was drawn at random according to an empirically observed distribution [50], then scaled with the factors t equal to $0, 1, 2, 4$, and 10 before obtaining the required number of transmitters per cell. The resulting average numbers of transmitters per cell are respectively 1.00, 1.05, 1.17, 1.47, and 2.20, respectively. The available spectrum consists of 75 contiguous frequencies.

Siemens Siemens AG provided four different benchmarks:

Siemens1 A GSM 900 network with 179 active sites, 506 cells, and an average of 1.84 transmitters per cell. The available spectrum consists of two blocks containing respectively 20 and 23 frequencies.

Siemens2 A GSM 900 network with 86 active sites, 254 cells, and an average of 3.85 transmitters per cell. The available spectrum consists of two blocks containing respectively 4 and 72 frequencies.

Siemens3 A GSM 900 network with 366 active sites, 894 cells, and an average of 1.82 transmitters per cell. The available spectrum comprises 55 contiguous frequencies.

Siemens4 A GSM 900 network with 276 active sites, 760 cells, and an average of 3.66 transmitters per cell. The available spectrum comprises 39 contiguous frequencies.

Swisscom Modified (*provided by Swisscom Ltd.*) A GSM 900 network in a city with many locally blocked channels which simulates a partial assignment that has to be completed. The average number of transmitters per cell is 2.09. There are 148 cells with 1 to 4 transmitters and 707 neighbour relations. 3, 49 frequencies are available, but 136 cells have restrictions, with the worst case of only 10 frequencies available. The median of available frequencies per cell is 19.

K (*provided by E-Plus Mobilfunk GmbH*) Data for dense urban environment in a GSM 1800 network with only 264 cells and 267 transmitters, with an average number of 151 interference relations per transmitters.

In the following we will present other characteristics of the COST259 benchmarks considered. Firstly, Table 4.6 gives for each of them the number of sites, the number of cells, the average and maximum number of transmitters per cell, and the size of the frequency spectrum (or the sizes of the contiguous portions if there are globally blocked channels). Note that in the latter case the gap in the spectrum is always greater than the maximum required separation thus no interference will ever occur between separated blocks of the spectrum. In addition, two instances, that is Siemens3 and Swisscom, present locally blocked channels. Moreover, Siemens3, Swisscom Modified, and Bradford1 present disconnected components. The reasons of this are not entirely clarified, as reported in [40]. However, the only numerically relevant component belongs to Siemens3 (36 vertices) whereas for the other two data sets they are actually constituted by a few isolated vertices.

Table 4.7 shows the characteristics of the corresponding interference graphs $G(V, E)$. Namely, we are showing the number of vertices $|V|$, the density of the graph, the average and maximum degree, the size of the maximum level zero clique, and information concerning the number, the average, and the max values of the soft and hard constraints. The *density* of the graph is expressed by the ratio between the total number of edges, either representing hard or soft constraints, and the number of edges corresponding to a complete graph.

Table 4.6: COST259 scenario characteristics [40]

	sites no.	cells no.	Mean trans\cell	max. trans\cell	spectrum size
brad0	649	1866	1.00	1	75
brad1	649	1866	1.05	3	75
brad2	649	1866	1.17	5	75
brad4	649	1866	1.47	9	75
brad10	649	1866	2.20	12	75
siem1	179	506	1.84	4	20, 23
siem2	86	254	3.85	6	4, 72
siem3	366	894	1.82	3	55
siem4	276	760	3.66	5	39
Swisscom Modified	87	148	2.09	4	3, 49
k	92	264	1.01	2	50

Table 4.7: Graph characteristics of the COST259 benchmarks [40]

	$\|V\|$	density %	Mean degree	max degree	max clique
brad0	1886	13.59	256.4	779	81
brad1	1971	13.46	265.3	805	84
brad2	2214	13.5	299.0	916	93
brad4	2775	13.44	373.0	1133	120
brad10	4145	13.41	555.9	1704	174
siem1	930	9.03	84.0	209	52
siem2	977	49.17	480.4	877	182
siem3	1623	9.18	149.1	519	78
siem4	2785	10.5	292.3	752	100
Swisscom Modified	310	8.29	25.7	94	21
k	267	56.57	151.0	238	69

Several considerations about the graph characteristics are worth noting:

- Two problems (Siemens2 and K), present a graph density widely above the average (close to 50% in both cases). In particular, test problem K aims to simulate a network situated in a very dense urban area. Intuitively, high values of potential interference make the decomposition approach less likely to be effective when used to solve FAP instances.

	hard const. no	co-ch. no	Mean co-ch	max co-ch	adj-ch. no	Mean adj-ch	max adj-ch
brad0	7288	234479	0.09	1.8	4263	0.02	0.8
brad1	7996	253441	0.09	1.8	4825	0.02	0.8
brad2	10284	320684	0.09	1.8	6871	0.03	0.8
brad4	16663	500805	0.09	1.8	12524	0.04	0.8
brad10	38234	1113850	0.09	1.8	33548	0.05	0.8
siem1	6039	33002	0.07	1.7	9911	0.02	0.6
siem2	17761	216912	0.02	0.5	25615	0.00	0.0
siem3	23093	97861	0.03	1.1	15069	0.01	0.3
siem4	27964	379052	0.03	0.7	26445	0.01	0.0
Swisscom	3984	0	0.00	0.0	2075	0.29	1.0
k	1053	19111	0.15	1.9	996	0.03	0.8

- All the data sets present both co-channel and adjacent-channel interference as soft constraints, with the exception of *Swisscom Modified* in which only the latter is existing and also shows interference values much greater on average than in the other benchmarks. All co-channel constraints are considered as hard constraints.

- The average degree is significantly higher than the number of channel available in the spectrum, with *Swisscom Modified* being again the only exception. This implies that the frequency assignment between adjacent cells has to be suitably tuned in order to avoid high values of the interference [40].

- Finally, the maximal clique size is also greater than the spectrum size for all data sets, although it is a low percentage of the cardinality of the corresponding graph. This means that no feasible assignments, that is those respecting the hard constraints, can be interference free [40].

Cardiff University data sets

Although the COST259 are the most realistic class of publically available benchmarks for the MI-FAP problem, their sizes, from one to few thousand of transmitters, is still much lower than those we can find in real life instances. It is important to test the effectiveness of the decomposition approach on larger data sets, since we know that the performance of the standard meta-heuristics degrades rapidly with the increasing problem size. For this reason, in order to complete the set of MI-FAP data sets, two larger problems C_5 and C_6 in the order of $10,000$ transmitters have been artificially generated. The idea behind their generation was to extend one of the COST259 test problems to a much larger size, with the aim of conserving its local graph structure characteristics but 'spreading' it at the same time over a much larger area.

We focus our attention on the first of the Siemens benchmarks, which consists of $c = 506$ cells distributed over an area of $xy = 240 * 125$ in cartesian coordinates, with a total of 930 transmitters. Hence the cell density d is equal to

$$d = \frac{c}{xy} = \frac{506}{240 * 125}.$$

We assume a number of 2500 cells in the new problem (equal to five times that of Siemens1) and distributed over a square region of side x'. The cell density d' of the new problem is then expressed by:

$$d' = \frac{2500}{(x')^2}.$$

To retain the same cell density (i.e. $d = d'$) we require:

$$x' = \sqrt{\frac{x * y * 2500}{c}} = 430. \qquad (4.2)$$

To generate the new data sets we applied the *benchmark generator* tool described in [9] for the MS-FAP benchmarks to produce cell locations. We have considered two different gaussian probabilistic distributions centered respectively on one and two towns over the square region in order to produce approximatively the same number of cells. In detail, we have considered an area of 430*430 in cartesian coordinates and centered the 'single town' distribution on the location (250,250) for a spread length of 200. The other parameters which define the gaussian distribution were a height of 0.02 and a cut-off of 0.001, as recommended in [9] as the most typical values. Similarly for the for the 'two towns' distribution we had two distinct centers at the locations (100,300) and (300,100) with an equal spread length of 150. The values for the distribution height and cut-off were respectively 0.04 and 0.007. Note that these parameters are higher than those adopted for the one town example in order to produce approximatively the same number of 2500 cells. We obtained 2730 and 2871 cells for the single and two towns problems respectively. Transmitters locations are shown in Figure 4.7.

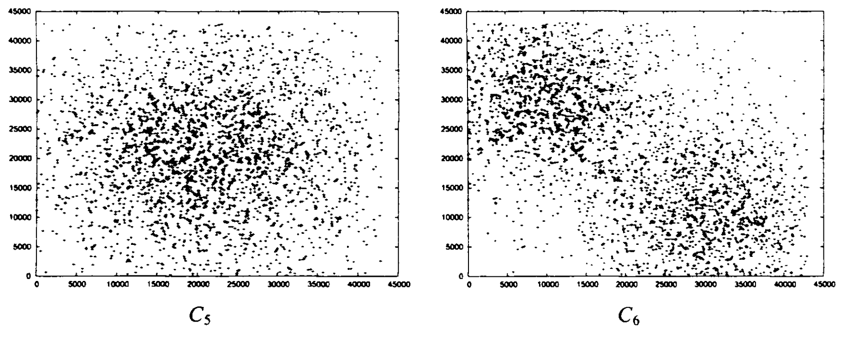


Figure 4.7: Cells locations for data sets C_5 (top) and C_6(bottom) generated respectively with a 'single-town' and a 'two-towns' probability distribution

Subsequently, we generate the traffic demands of each of the cells by assigning a number of transmitters drawn following a uniform distribution between one and seven to obtain a total number of transmitters in the order of 10,000. This corresponds to an average of four transmitters per cell approximatively. Note that this average value is approximatively equal to those in the harder Siemens benchmarks, i.e. Siemens2 and Siemens4.

We consider one cell per site and ignore the interference due to the hand-over. Consequently, the hard constraints consists only of three channel separations between transmitters within the same cell. To determine the interference values for the definition of the soft constraints we have again referred directly to the Siemens1 problem by applying the following procedure. Firstly, we plot in two separate graphs all the co-channel and adjacent-cannel interference values as a function of the Euclidean distance between pairs of transmitters (Figure 4.8). Note that we consider all the occurrences in Siemens1, thus including the pairs which do not produce any interference, which are assigned an artificial value of zero. Subsequently, for each pair of transmitters in C_5 and C_6 we compute their Euclidean distance and then assign to the corresponding edge a value of the co-channel interference drawn at random among all the occurrences displayed in Figure 4.8 at the same distance within a small range. In a similar way we determine the adjacent-interference values. Finally, for both problems we have used the same frequency spectrum provided for the Siemens1 instance. Table 4.8 shows the characteristics for the two additional large MI-FAP benchmarks C_5 and C_6 generated by the procedure described above.

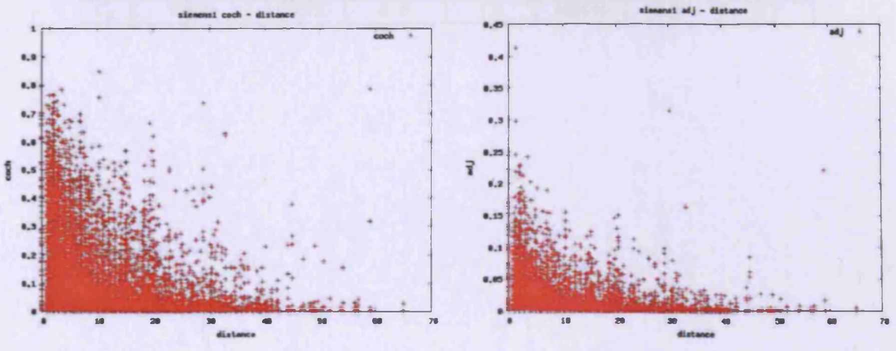

co-channel adjacent-channel

Figure 4.8: Interference-distance plot for co-channel and adjacent channel interference in Siemens1

We can observe from these tables that the graph density is considerably lower than in the Siemens instances (and that in Siemens1 in particular). However, this density refers to the whole extended area of the newly generated benchmarks, whereas we will be more interested in having the 'local' edge densities close to each other. In fact, we can expect the new extended problem as having approximatively the same density of the Siemens1 problem when this is calculated over areas having equal or smaller size of the original problem itself. Moreover, the comparison should be conducted at different locations, in order to distinguish between high density areas, such as the centers of the 'towns', and regions with less connected transmitters.

Table 4.8: Characteristics of benchmarks C_5 and C_6

	sites no.	cells no.	Mean trans\cell	max. trans\cell	spectrum size
C_5	2730	2730	4.01	7	20, 23
C_6	2871	2871	4.14	7	20, 23

	$\|V\|$	density %	Mean degree	Max degree	Max clique
C_5	10935	0.0176	100.47	230	28
C_6	11519	0.0177	106.32	264	34

	hard const. no	co-ch. no	Mean co-ch	max co-ch	adj-ch. no	Mean adj-ch	max adj-ch
C_5	21811	1010406	0.03	1.6	319042	0.01	0.3
C_6	22891	1132570	0.03	1.3	353890	0.01	0.3

Chapter 5

Minimum Span FAP

This section evaluates the decomposed solution approach on a number of MS-FAP test problems. This is the only FAP problem in which a decomposed approach has been previously used with meta-heuristics, by applying simulated annealing to cliques [62]. However, the proposed approach used differs substantially from the procedure in Algorithm 3.1 since it only assigns the clique of the graph at first (identified as the hardest part of the problem) and then tries to extend the resulting assignment to the whole graph (see Algorithm 3.5).

5.1 Heuristic algorithms

To test the validity of the decomposed assignment approach we apply to the MS-FAP the 'permutation-based' steady state genetic algorithm originally proposed by Valenzuela et al. in [119]. This algorithm obtained very good results on widely accepted test data sets for the MS-FAP.

Genetic Algorithms (GAs) are search methods originally developed by Holland [58] with the goal to either reproduce the natural process of evolution or to adapt it to design software systems retaining its original mechanism. Although GAs have been very scarcely used to solve the FAP, they are able to process a number of different solutions in parallel. Thus they can potentially explore a wider range of the search space than local search meta-heuristics, and potentially produce better or comparable results in a shorter time.

The 'permutation-based' GA incorporates a sequential procedure for individual fitness evaluation. The representation used (also called *order-based*) represents transmitters by finite sets of integers. Individuals in the population consist of orderings (permutations) of all the transmitters. Fitness is evaluated for each individual by applying a sequential constructive method. The implementation used for this thesis is the original SEAMO proposed in [118] with small modifications in the generation of the initial population and in the replacement mechanism. For a description of the SEAMO framework see Section 3.10 and Appendix C.2.2. The genetic operators used to generate offspring are *order based mutation* and *cycle crossover*. These were selected as the most effective recombination operators after some test runs [23]. The sequential procedure introduced by Hale in [53] is then used as fitness evaluation. Firstly, an ordering representing a single individual in the population is chosen among those introduced in Section 2, with the *Generalized Smallest Degree Last* (GSL) producing the best results in some test runs performed. This ordering has been used to create the initial population of individuals by applying the crossover operator to this ordering and other randomly generated permutations. The GLS algorithm is briefly described in the following:

- Given an initial random order of transmitters the transmitter having smallest degree is selected and added to a list, breaking ties by selecting transmitters which appear firstly in the initial given ordering. Then the selected transmitters is removed from the initial set. Subsequently, the selection procedure repeated for the updated set of transmitters, and the new transmitter added to the list. When all of the transmitters are selected the list is reversed returning the final ordering.

The following procedures are applied in sequence for the channel assignment: the selection of the next transmitters is implemented as sequential (that is by simply selecting the next transmitter in the ordering), then a frequency is assigned to it with the simplest Smallest Acceptable Frequency algorithm (SAF) [54] (the smallest frequency which can be assigned without violating any of the constraints involving the selected transmitter). Algorithms 5.1 and 5.2, give respectively the pseudocode of the order-based GA and the sequential assignment used as fitness evaluation.

It can be observed that the sequential procedure always produces assignments which are zero-violation assignments, which makes the algorithm particularly suitable for the MS-FAP. Furthermore, it has been proved that the permutation-based representation is able to represent any possible solution in the objective space (that is a given frequency assignment to a set of transmitters), thus permitting full exploration of the whole solution space [118]. This is the reason why the mutation

Algorithm 5.1 Pseudocode of the 'permutation-based' GA (SEAMO)

```
1: Generate a population of $popSize$ individuals as permutations of the whole set of transmitters representing a chromosome.
2: Evaluate the fitness for each individual by using a Sequential algorithm to generate an assignment
3: Store the $bestSoFar$ fitness value
4: while Stopping condition not satisfied do
5:    while next individual in the population do
6:       { This individual becomes the first parent }
7:       { Select a second parent either at random or by applying roulette wheel selection }
8:       { Apply crossover to produce offspring }
9:       { Apply mutation to offspring }
10:      { Evaluate fitness produced by offspring }
11:      { Compute and add fitness sharing terms. }
12:      if offspring better than either parent then
13:         { Replace the weaker parent }
14:      else
15:         if offspring fitter than weakest individual in the population then
16:            { Replace another weaker individual in the population selected at random }
17:         end if
18:      else
19:         Discard offspring
20:      end if
21:      { Update $bestSoFar$ }
22:   end while
23: end while
24: Select the ordering representing the $bestSoFar$ individual
25: Assign channels to it using a Sequential algorithm
```

operator adopted in this implementation is very basic (it only consists of swapping two transmitters at random in an ordering) whereas in most of the GAs proposed in the literature it has the function of a proper local search, which generally aims to prevent the algorithm from converging to local minima.

Moreover, the order based GA can be also potentially effective in the other FAP problems proposed which are seeking to produce assignments (even if partial ones with some of the transmitters left unassigned). In particular, although not considered for this thesis, it would be interesting to apply our algorithm to the

Algorithm 5.2 Fitness evaluation for the MS-FAP

```
Input:   Chromosome ord
Output   FrequencyAssignment $f$, Int $span\text{-}O_{MS}$
1: first transmitter in ord $\leftarrow f_0$
2: while next transmitter in ord do
3:     { Select next unassigned transmitter $x$ in the initial ordering ord }
4:     { Select the lowest available frequency $f_x$ which can be assigned to $x$ without violating
       any constraints with the transmitters already assigned in ord }
5:     { $x \leftarrow f_x$ }
6: end while
7: Evaluate the $span - O_{MS}$ of $f$
8: return $f$, $span - O_{MS}$
```

Maximum Service FAP described in Section 1.4. Here, because of the sequential procedure incorporated the order based GA should solve the problem of maximize the number services naturally.

The parameter setting used is that originally proposed in [118] and successively used in [25]. Each of the experiments has been run for 500 generations with an initial population size of 1000 chromosomes. The roulette wheel scheme has been adopted in the selection process, cycle crossover has been applied at a rate of 100% and one order based mutation for each individual have been used. For more details about the genetic operators used see Section 3.10 and Appendix C.1. Only one loop has been conducted over each of the given partitions. Since the representation used allows full coverage of the search space no further local search needs to be added after the evaluation of each offspring. Consequently, only one evaluation is needed for each new individual in the population and for each generation. Hence, with the parameter values above, this corresponds to approximatively $500 \times 1000 = 500,000$ evaluations. The number of evaluations computed for each of the subsets is split proportionally to the number of vertices contained in each subset according to Definition 4.1.

5.2 Decomposition algorithms

In order to simplify we have restricted the type of decompositions tested to those previously used in the literature, see [62], and in preliminary experiments published in [24, 25]. In particular we have tested geographical, clique, generalized-degree, and the simplest implementation of the GPFAP decomposition described in Problem 3.3.

Generalized degree decomposition has been implemented according to Algorithm 3.9. GSL has been selected as the algorithm used to produce the initial ordering as it produced the best overall performance on a number of random sequential assignments for the majority of the data sets tested. Furthermore, the decomposition method in Algorithm 3.9, based on the area under the plot generalized degree versus transmitters, has been preferred to the other option of decomposing proportionally to the number of transmitters contained into each subset, since it appeared superior in all the preliminary experiments performed in [25].

Clique decomposition has been used in the basic unweighted version proposed in Algorithm 3.6 and previously used as the 'level zero' clique in [111]. However, our implementation presents marked differences in the construction of the next subsets once the maximum clique of the interference graph has been found (see Algorithm 3.8). In fact, in the approach used in [111] the first cliques were only further extended whenever the difference between the span produced by them and that of the completed graph was above an arbitrarily fixed threshold and this procedure is not able to set a priori a desired number of subsets. On the contrary, we aim to find a procedure which is capable of investigating constructively the potential of each decomposition method, without any consideration of the cost and number of subsets.

We are expecting a decomposition to be successful when the span produced by solving the subsets independently is close to that of the final assignment. If this happens for any of the subsets, the corresponding partial assignment has good chances to be extended to a nearly optimal assignment of the whole set of transmitters.

Graph partitioning decomposition has been applied using the basic cost in Problem 3.3 which minimizes all of the inter-edges between pairs of different subsets. In addition, only the balanced GPFAP has been considered for this problem. For both clique and graph partitioning preliminary tests have shown that better results are obtained whenever the transmitters contained in each of the subsets are re-ordered according to GLS. This is the ordering (chosen among those proposed in [54]) which performs better in a number tests conducted by performing simple sequential algorithms on the same MS-FAP benchmarks. As a consequence the performance of the GA (which involves a sequential assignment as fitness evaluation) depends strongly on the specific ordering by which transmitters are selected within each of the subset, as random orderings produce generally very poor performance

For data sets C_1, ... , C_4 information specifying the location of the transmitters is provided. Therefore, it is possible to apply the geographical decomposition in Algorithm 3.4. For these experiments the parameter D is assigned the value of half the range of the x coordinate to determine transmitters to be included respectively in the first and second subset. Any transmitters which have not been assigned to either of the two subsets are included in the first subset. The procedure is applied recursively to produce decompositions into more than two subsets.

This simple geographical procedure is probably the most intuitive decomposition algorithm we can imagine, if we think in terms of clusters of transmitters in the network. Moreover, for the problems generated by a 'two towns' distribution the partitioning produced by the GPFAP and geographical decomposition appear very close. This is for example shown by Figure 5.1, which visualizes the results of the decomposition for test problem C_4. The plot clearly shows that with graph partitioning transmitters belonging to each of the two different towns are largely included into distinct subsets. On the contrary, a different partition is produced by generalized-degree, in which is instead emphasized the idea of including the most highly connected vertices in one of the subsets only. Finally, a single random decomposition has been tested for comparison, using the implementation described in Algorithm 3.3.

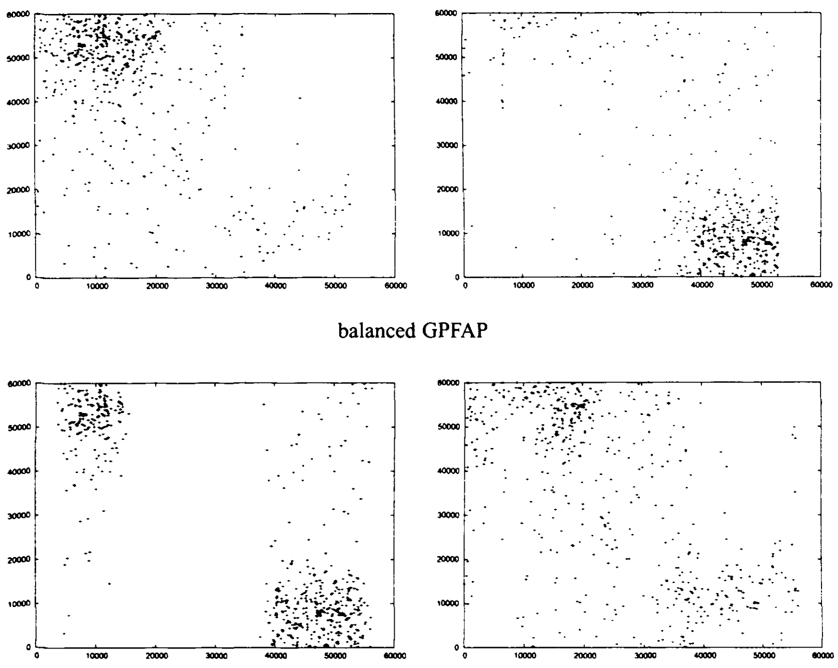


Figure 5.1: Decomposition into two subsets for data set C_4

5.3 Experimental results

Table 5.1 gives the outcomes for all the benchmarks described in 4.2.1 in terms of best and mean span (shown in brackets) over three runs of the assignment with the heuristic with different random seeds. The values in bold indicate the best span produced by either approaches.

All the results produced by the decomposition approach were obtained by solving the subsets sequentially, respecting the constraints with those already assigned, since solving in parallel will inevitably produce constraint violations. A comparative percentage analysis of the different decomposition methods used can be found in Appendix A.

By analyzing the results in Table 5.1 we can make the following observations. For test problems (P_1,P_2) the GA with decomposition finds the optimal solution with either the decomposed assignment (GPFAP and generalized degree) or the whole approach.

For the other test problems the generalized degree algorithm appears superior to the others and the decomposition approach is always successful in all the instances tested. That is it produces a lower span than the non-decomposed approach. The GA can be compared to other local search heuristics when used in the traditional whole approach, such as tabu search and simulated annealing in their standard implementations proposed in [62] for the software tool FASoft. This is one of the most fully documented systems for the FAP problem which implements a number of different meta-heuristics available to solve FAP instances. A comparison between the GA and other algorithms results can be found in [24,25].

GPFAP and clique, although producing competitive results in terms of good approximation of the optimal solution, do not generally find a better span than the non-decomposed approach. Clique decomposition seems to be slightly superior to the balanced partitioning GPFAP. This can be explained by the the specific cost used for this problem, that is the span of an assignment. In fact, the first cliques in the sequence of subsets are generally able to produce costs which are reasonably close to the span of the final (nearly) optimal assignment of all the transmitters in the network. This happens in particular for the first subset which is always composed by the maximum level zero clique. Note that for these benchmarks, because of either the relatively small size and low graph density, the maximum clique can be found with a reasonable computational effort.

GPFAP appears less effective on average than the generalized degree and clique decomposition. However there are few exceptions. In the case of the Philadelphia

Table 5.1: Best and mean span for the MS-FAP test problems solved by order-based GA with decomposition (500,000 evaluations)

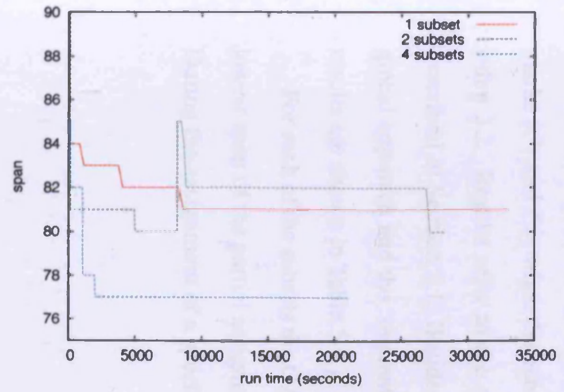
ns	Random	Geog.	GPFAP	Gen. Degree	Cliques
			P_1		
1	**47 (47.3)**				
2	51 (51.3)	–	**47 (47.3)**	**47 (47.3)**	**47** (47.7)
4	51 (51.7)	–	48 (48.3)	**47** (47.7)	48 (48.0)
			P_2		
1	**426** (426.7)				
2	479 (480.3)	–	**426 (426.3)**	**426** (426.7)	430 (432.0)
4	543 (545.3)	–	427 (428.0)	428 (428.3)	431 (433.3)
			C_1		
1	81 (81.7)				
2	100 (102.3)	94 (94.3)	83 (83.7)	80 (80.7)	82 (82.3)
4	106 (104.0)	92 (93.3)	86 (87.0)	**79 (80.3)**	84 (84.7)
			C_2		
1	74 (74.7)				
2	85 (86.7)	84 (84.7)	81 (82.7)	74 (74.7)	77 (77.7)
4	88 (90.0)	85 (86.7)	79 (80.3)	**73 (73.3)**	78 (79.0)
			C_3		
1	74 (75.0)				
2	89 (90.7)	82 (82.0)	78 (78.3)	74 (75.0)	79 (79.7)
4	90 (91.0)	84 (84.3)	79 (79.0)	**73 (74.6)**	77 (77.3)
			C_4		
1	75 (75.3)				
2	98 (99.3)	71 (71.7)	**69 (70.3)**	73 (74.0)	73 (74.0)
4	106 (106.7)	73 (74.3)	74 (74.3)	71 (71.3)	72 (73.3)
			G_1		
1	54 (56.0)				
2	56 (58.0)	–	53 (53.3)	53 (54.7)	53 (53.3)
4	57 (57.7)	–	56 (57.0)	**51 (52.6)**	55 (55.0)
			G_2		
1	55 (57.3)				
2	66 (67.3)	–	58 (58.0)	55 (56.3)	55 (56.0)
4	67 (68.0)	–	**54 (54.3)**	**54** (55.7)	55 (55.7)
			G_3		
1	**106** (106.3)				
2	115 (116.0)	–	107 (107.3)	**106 (106.0)**	**106** (106.3)
4	119 (120.3)	–	110 (110.7)	107 (107.6)	**106** (106.7)
			R_1		
1	121 (122.3)				
2	144 (145.3)	–	125 (125.3)	**120 (121.7)**	123 (123.7)
4	141 (141.7)	–	127 (128.7)	121 (122.3)	124 (125.3)
			R_2		
1	**216** (217.0)				
2	294 (295.0)	–	225 (225.7)	**216** (216.7)	219 (219.3)
4	308 (308.7)	–	227 (227.3)	**216 (216.3)**	218 (218.7)

instance P_2 the GPFAP appears to be the best performing decomposition method whereas clique decomposition is particularly penalised. This can be in part explained by the nature of these benchmarks, which presents a very high and unrealistic cell demand. More importantly, the GPFAP appears superior to the other methods and the non-decomposed approach for data set problem C_4, since this represents an intuitive and natural form of partitioning for these benchmarks in which transmitters are located in 'different towns'. Note that for this benchmark the traditional approach of solving the problem as a whole appears particularly penalised.

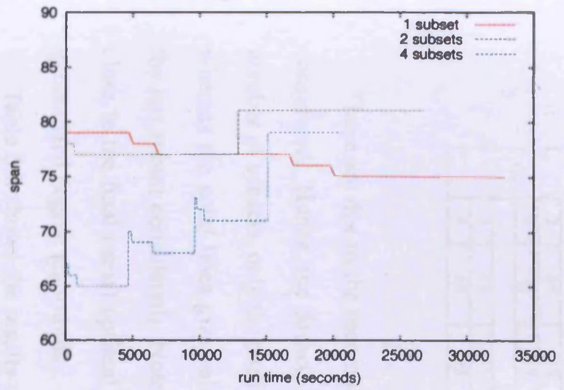
As expected random decomposition does not produce good results, while the geographical decomposition results are not satisfactory in the case of the 'one-town' benchmarks C_1 and C_2. Furthermore, for the problems generated with a distribution in 'two-towns' this method still offers no advantages over GPFAP.

Despite the peculiarity of their structure (which presents a very high and unrealistic graph density), the outcomes for the random graphs confirm the results obtained for the other MS-FAP benchmarks. In particular, generalized degree appears slightly superior to clique decomposition and GPFAP, although they equalize its results in some problems. Note that, although the decomposed assignment approach matches the results obtained with the undecomposed solution, because of the very high edge density of the graph, the gain in runtime for these problems can be large.

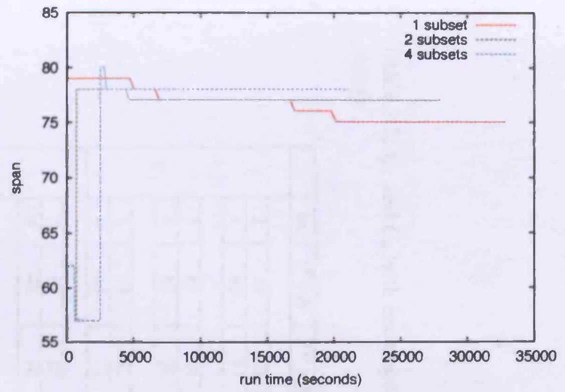
Figure 5.2 shows span-time plots for a number of specific runs of the benchmarks in Table 4.1 solved by the different decomposition algorithms after $500,000$ total evaluations. The diagrams show the respective behaviours when these problems are solved as a whole, and with the decomposition technique. To simplify we only show the results produced by a partition into two and four subsets. As expected the *span* produced by a particular subset can be lower than the final *span*, and some peaks appear in the diagram.



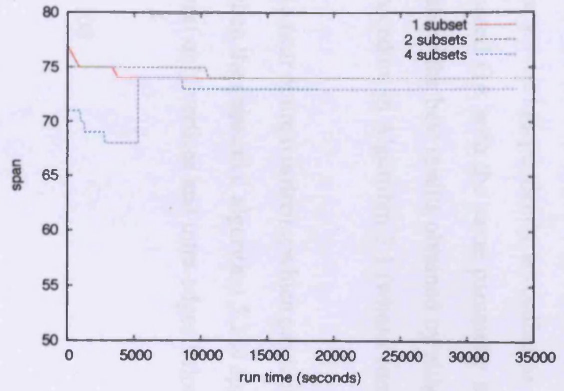
C_1 - generalized degree



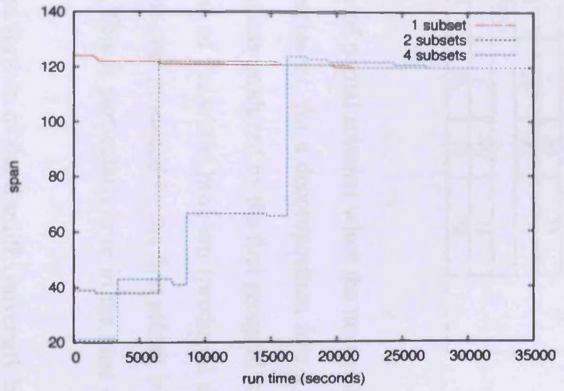
C_2 - GPFAP



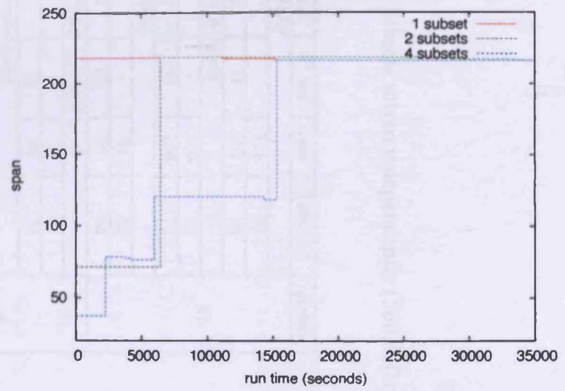
C_2 - clique



C_3 - generalized degree



R_1 - generalized degree



R_2 - generalized degree

Figure 5.2: Span-Time plot for the MS-FAP benchmarks solved by the order-based GA with decomposition into two and four subsets

Table 5.2: C_1 and C_4 with decomposition - subsets solved independently (500,000 evaluations)

	ns	1st sub	2nd sub	3rd sub	4th sub	cost	Table 5.1	global
C_1	GPFAP							81
	2	39	63	-	-	118	83	
	4	62	61	42	31	102	86	
	Generalized-degree							
	2	63	49	-	-	110	80	
	4	62	50	43	35	100	79	
	Clique							
	2	62	63	-	-	94	82	
	4	62	23	14	63	95	84	
C_4	GPFAP							75
	2	61	52	-	-	98	69	
	4	44	40	16	29	90	74	
	Generalized-degree							
	2	65	53	-	-	89	73	
	4	46	37	24	24	82	71	
	Clique							
	2	65	71	-	-	90	73	
	4	65	23	15	68	93	72	

These are due to the recombination of partial solution when the next subset is considered. Hence, the dashed lines represent, for a decomposition into a given number of subsets, only the partial solutions produced by the first group of subsets whereas the solid lines give valid spans of the whole problem (produced during the last subset considered). Note that the cost produced by the first subset is often close to the final (near) optimal span. This is particularly true in the case of the generalized degree algorithms.

Table 5.2 shows the results obtained for two of the Cardiff University benchmarks (C_1 and C_4) when the subsets are solved independently according to Algorithm 3.2. Results refer to the order based GA with the same parameter setting described in Section 5.1. Besides, we show the best results obtained by either the global approach and the 'sequential' procedure in Algorithm 3.1 (whose complete results are shown in Table 5.1)

For each of the subsets the GA finds a near optimal ordering which produces the lowest span of the partial assignment when the sequential algorithm 5.2 is applied. During the assignment of a specific subset only vertices and intra-edges belonging to it are considered. Then the best ordering is returned by the heuristic for each subset.

When all the subsets have been solved the corresponding partial orderings are concatenated together sequentially (according to the original subsets ordering). Then a final complete assignment of the whole set of transmitters is generated by simply applying once Algorithm 5.2 to the merged ordering (reconstruction). This will only add one further evaluation to the decomposed procedure (produced by the application of the sequential assignment 5.2), thus not being significant with concern to the count of total evaluations, see also Section 4.1.1. Note that apart from the first subset in the list of subsets which keep the same assignment, all the assignments for the other subsets will change during the reconstruction procedure. Note that this further reconstructions of partial solutions in order to produce the final assignment is not needed when the subsets are processed sequentially since the procedure directly returns a final valid assignment.

We present the best span over three runs, together with the partial spans produced by each single subset. We have used GPFAP, generalized-degree and clique decomposition since they are the best performing methods for the results in Table 5.1 obtained with the 'sequential' decomposed assignment technique.

None of the decomposition methods produces satisfactory results when the solutions are recomposed together for both test problems considered. This is a consequence of the fact that the 'partial' orderings obtained when the subsets are solved independently ignore the inter-connection between subsets thus, when they are merged together, do not produce an optimal ordering for the complete graph. Clique and generalized-degree show a significant value of the span during the first subset. This result confirms the expectation since all the decomposition used aim to present the 'hardest' part of the problem first. The span generated for the remaining subsets is in general lower. However, for the clique decomposition the last subsets (which includes a significant number of vertices due to the small size of the cliques) still produces a high span. GPFAP with the cost defined as in Problem 3.3

aims to balance the subsets which then tend to produce a similar span. However, this tendency is more important for C_4 with two subsets, for which the GPFAP generates a decomposition into two distinct towns of almost same size.

In conclusion the MS-FAP results show that the decomposed assignment approach with the subsets solved sequentially improves or equalizes those obtained by the whole approach for all of the benchmarks tested. Decomposition methods that aim to isolate and solve at first the possibly hardest part of the data set (i.e. clique and generalized degree) produce the best performance, thus confirming some results previously obtained for this problem. To summarise Table 5.3 shows the best and average results obtained by the permutation based GA using both the global and the decomposition approach in comparison with the best results produced by simulated annealing (with global approach) as previously published in [25].

Table 5.3: Comparison between the best and average (in brackets) spans produced by the order-based GA and SA [25] ($500,000*$ evaluations)

benchmark	GA with decomposition	GA global	SA from [25]
P_1	**47 (47.3)**	**47 (47.3)**	**47 (47.3)**
P_2	**426 (426.3)**	**426** (426.7)	**426** (426.7)
C_1	**79 (80.3)**	81 (81.7)	81 (82.4)
C_2	**73 (73.3)**	74 (74.7)	74 (74.8)
C_3	**73 (74.6)**	74 (75.0)	76 (77.4)
C_4	**71 (71.3)**	75 (75.3)	77 (78.0)
G_1	**51 (52.6)**	54 (56.0)	54 (55.2)
G_2	**54 (54.3)**	55 (57.3)	55 (58.0)
G_3	**106 (106.0)**	**106** (106.3)	**106** (106.5)
R_1	**120 (121.7)**	121 (122.3)	121 (122.0)
R_2	**216 (216.3)**	**216** (217.0)	**216 (216.3)**

In the remaining part of this thesis we will apply the decomposed approach to the more complex FS/MI-FAP, for which it has never been used in previously published works.

Chapter 6

Fixed Spectrum FAP

This section applies the decomposition technique to a selection of FS-FAP test problems. In order to compare the results with those produced for the MS-FAP we apply the same heuristic used for that problem, that is the order-based GA. However, this needs to be adapted for the FS-FAP as described in the next section.

6.1 Heuristic algorithms

One advantage of the permutation based GA in Algorithm 5.1 applied to the MS-FAP was the property of generating only zero-violations assignments, thus reducing significantly the search space and speeding up its convergence without preventing the full exploration of the solution space. For the FS-FAP the domain of available frequencies is limited thus it is usually not possible to find a zero-violation assignment for a given set of transmitters. Therefore some interference becomes unavoidable. Furthermore, the algorithm which produces the assignment itself needs to be modified. We keep the same general structure of a sequential algorithm but the selection of frequency assigned to the next transmitter in the ordering is now modified according to Algorithm 6.1. The modified procedure assigns to the next transmitter the smallest frequency available in the domain which minimizes the costs O_{FS} defined in Problem 1.2 calculated on the transmitters already assigned in the ordering. It is important to mention that we can no longer give an equivalent of the proof in [118] and, as a consequence, guarantee the complete exploration of the search space.

To allow full exploration, Local Search (LS) can be incorporated into the GA after the application of the genetic operators, to explore those assignments which may not be found by the sequential algorithm. The LS implemented is simple, in order to preserve the structure and the effectiveness of the GA itself, with a single move defined as a single frequency change ($\pm$ one channel) for a number of transmitters selected at random in a given assignment to the whole network. The LS also stores and updates the best configuration obtained, thus the new assignment

Algorithm 6.1 Fitness evaluation for the FS-FAP

```
Input:   Chromosome ord
Output   FrequencyAssignment f, Int cost
1: first transmitter in ord $\leftarrow f_0$
2: while next transmitter in ord do
3:     { Select next unassigned transmitter $x$ in the initial ordering ord }
4:     { Select the lowest available frequency $f_x$ which minimize the sum of violations
       $\varphi_{FS}(f, e)$ with the transmitters already assigned in ord, }
5:     { $x \leftarrow f_x$ }
6: end while
7: Evaluate the cost as the sum of violations in $f$
8: return $f$, cost
```

can only have a cost which is better than or equal to the initial one produced by the sequential algorithm. Pseudocode of the fitness evaluation is shown in Algorithm 6.2. Two acceptance criteria have been used:

Definition 6.3 *Given the LS procedure in Algorithm 6.2 we define the two following acceptance criteria:*

- Hill Climbing *(see [63]): a move is accepted only if the cost of the new configuration is better than the old one*
- Metropolis *(see [84]): a move can be accepted even if it produces a greater cost O_{FS}, according to a probability distribution (6.1). The parameter k has been set to 1 while the temperature T depends on the particular instance considered and it is usually set in the range 0 to 10.*

$$P(O_{FS\,old}, O_{FS\,new}) = \begin{cases} e^{-\frac{O_{FS\,new} - O_{FS\,old}}{kT}} & if\ O_{FS\,new} > O_{FS\,old} \\ 1 & otherwise \end{cases} \tag{6.1}$$

Preliminary tests were needed in order to set the parameters for this modified version of the order-based GA (see Appendix C.1). However, no significant improvements were observed by changing the settings chosen for the MS-FAP and

Algorithm 6.2 Local search implementation for FS-FAP

```
Procedure Local Search for FAP
  Input:   Chromosome ord
           FrequencyAssignment f
  Output   FrequencyAssignment bestf
 1: Apply Sequential assignment to ord and set it as the initial value for f
 2: while  numberOfMoves < 10*numberOfTransmitters  do
 3:    { Select a transmitter x in ord at random }
 4:    { Store the frequency $f_x$ assigned to x in f }
 5:    { Select a new frequency $f_{xNew}$ at random as either $f_x + 1$ or $f_x - 1$ }
 6:    { $x \leftarrow f_{xNew}$ to obtain a new configuration $f_{xNew}$ }
 7:    { Evaluate the cost $0_{FS}$ of $f_{xNew}$ }
 8:    if  Move accepted  then { $f \leftarrow f_{xNew}$ }
 9:    if $0_{FS}$ < bestCost then
10:       { bestf $\leftarrow f_{xNew}$ ; bestCost $\leftarrow 0_{FS}$ }
11:    end if
12: end while
13: return bestf
```

reported in Section 5.1, which, therefore, has been maintained for the FS-FAP experiments too. In addition, LS was added with a Metropolis acceptance criterion with a variable value of T decreasing from a high value to a low value, set respectively to 10 and 0 in our tests (T 10 $\rightarrow$ 0). It is important to note that, although decomposition is effective and produces better results than the whole approach (at least with the GPFAP), the benchmarks used have on average a low connectivity (with the exceptions of the random graphs and the Philadelphia benchmarks), thus they do not represent particularly hard FS-FAP instances.

For harder fixed spectrum instances the order-based GA is not able to produce satisfactory results. For example with the last group of benchmarks G_1, G_2 and G_3, although the decomposition improves markedly its results, the order based GA is outperformed by other standard meta-heuristics using the whole approach. For a comparison with a standard SA see [25]. This is essentially due to the intrinsic limitations of the order-based representation and the impossibility of increasing the amount of the local search procedure without having the natural structure of the GA completely distorted. For this reasons we have conducted a number of experiments changing the type of the representation used and adopting the *direct representation* (see [23] for a comparison between different representations for GAs). Different genetic operators are needed for this representation, which will be also used for the MI-FAP described later (see also Appendix C.2.2). The crossover is applied with a rate of 100% and one *swap mutation* was executed per offspring. In addition, we applied the same local search Metropolis algorithm defined in 6.3 and already used for the order-based representation. In order to reduce the negative effect of the premature convergence *fitness sharing* has been added to the SEAMO structure of the GA with the implementation described in Section (3.6).

6.2 Decomposition algorithms

The FS-FAP instances have been tested with the decomposition methods already compared for the MS-FAP, namely GPFAP, generalized degree and clique. The cost for GPFAP is that used in Problem 3.3 which minimizes the inter-edges between subsets while maximizing the intra-edges in the first subset only.

Data sets G_2 and G_3, both present the characteristic of having *disconnected components* in the interference graph. For this reason, although the size of these components is very small, we have also considered the simple minimum-cut decomposition proposed in Problem 3.1.

Definition 6.4 *For the minimum-cut two different decompositions are proposed:*

- minCut1 (component). *This partitioning (which is automatically found by the min-cut procedure in Problem 3.1) is simply obtained by including in the first subset only the transmitters belonging to the minor disconnected components of the interference graph.*

- minCut2. *This partitioning is obtained iteratively by adding to the first component the transmitters included in the next minimum cut of the induced graph obtained by removing the vertices so far in the subset. The itera-*

tive procedure stops when a minimum number vertices has been reached, (5% in our experiments) has been reached. Note that this strategy becomes necessary because of the very small number of vertices included in each of the smallest sides of the next minimum cuts found (usually less than ten).

For *min-cut2* in order to obtain a partitioning into more than two subsets the procedure is applied recursively to each of the sides of the cut produced whereas for the 'component' approach outlined in *min-cut1* only a decomposition into two subsets has been considered.

6.3 Experimental results

Table 6.1 summarises the outcomes for the benchmarks in Table 4.1 using the order-based GA in Algorithm 5.1 with the modifications described in Section 6.1. We use the same decomposition criterion in Definition 4.1 in order to split the total number of evaluations (500,000 as well as for the MS-FAP) among different subsets. The cost used in Table 6.1 is that defined in Problem 1.2 for FS-FAP.

With exception of the first two Cardiff University instances (both generated with a 'one-town' distribution) the GPFAP is the most successful technique. In particular it appears very effective for data sets C_3 and C_4, which were generated with the benchmark generator tool using a 'two-town' distribution, and for the GSM scenarios G_1 and G_2. For this type of FAP the use of a decomposition method which aims to reduce the amount of inter-connections between different subsets is crucial, and the GPFAP directly targets this parameter as the objective of minimization. In particular, for the two-towns problem C_4 we can see from the plot in Figure 6.1 that in a decomposition into two subsets the cost produced by each of subsets are very close (and lower than that produced by the whole approach). This good performance could have been predicted since the decomposition produced by GPFAP into two distinct and nearly disconnected towns is naturally effective. Note that, this does not happen for the decomposition into four subsets, in which the cost in the diagrams presents a marked discontinuity between subsets whereas the first subset still produces only interference free assignments.

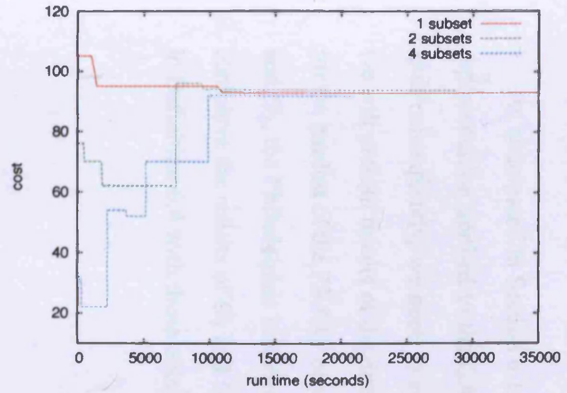
Clique decomposition appears far less effective than in the MS-FAP, and in only few of the instances it is actually able to improve the results produced by the non-decomposed approach. This can be explained by the fact that, for these data sets, the size of the cliques is very small compared to the whole graph. As a consequence, this decomposition method does not produce any effective partial costs for the first subset, or the first group of subsets. Hence the corresponding assignments are very often interference free.

Generalized degree decomposition is not as effective as it was for the MS-FAP. This can be explained by the fact that this method no longer takes advantage of the sequential assignment, which, although modified for this different problem, is not able to explore completely the search space. As a consequence a local search procedure needs to be added (see Figure 6.2) and this also limits the advantage of starting from a good ordering. In particular this method appears able to obtain some good results for the 'one-town' benchmark generated test problems, in which the approach of solving the mostly connected part of the graph at first is still effective. For the other test problems the requirement of minimizing the inter connections between subsets becomes predominant (especially in the 'two-towns' data sets), thus other decomposition methods outperform the generalized degree. Random and geographical decomposition produce results which are in percentage terms worse than those of the MS-FAP, when compared to the best costs found for each of the test problems.

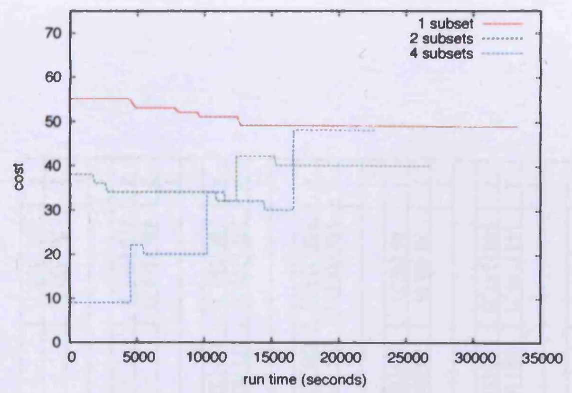
Figures 6.1 shows the cost-time plots for a number of specific runs of the benchmarks in in Table 4.1 solved by the different decomposition algorithms after $500,000$ total evaluations. The diagrams show the respective behaviours when these problems are solved as a whole, and decomposed into two and four subsets. Note that for the solutions obtained by the decomposed assignment approach only the solid line (which shows the last subset in a specific decomposition) represents a valid cost of a complete assignment of all transmitters in the network. It clearly appears that even a small number of subsets improves considerably the overall run time of the algorithm for the same effectiveness.

Table 6.2 shows the results obtained for benchmarks C_1 and C_4 when the subsets are solved independently according to Algorithm 3.2. For each of the subsets the GA finds the near optimal ordering which produces the lowest cost when the sequential algorithm 6.1 is applied. During the assignment of a specific subset only vertices and intra-edges belonging to it are considered. When all the subsets have been solved the partial orderings are concatenated together sequentially (according to the original subsets ordering), then a final complete assignment of the whole set of transmitters is generated by applying Algorithm 5.2 to the merged ordering. We present the best cost over three runs, together with the partial costs produced by each single subset, obtained with GPFAP, generalized-degree and clique decomposition, i.e. the best performing methods for the results in Table 6.1.

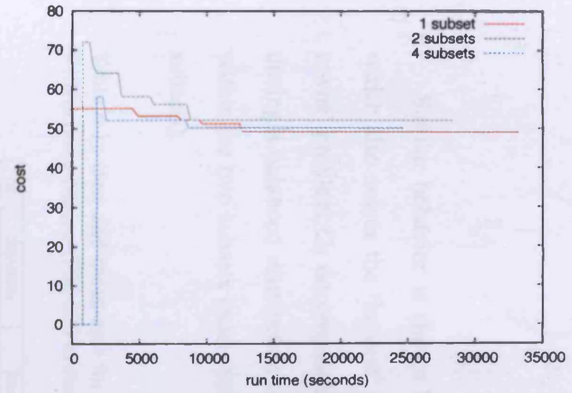
As for the MS-FAP (see Section 5.3) none of the decomposition methods produces satisfactory results when the solutions are recomposed for either of the test problems considered. From the partial costs results that clique decomposition is no longer capable of generating any significant values for the cost in the first, or first group of subsets. This is due to the smaller size of the cliques which tends to produce partial assignments which are interference free assignments, whereas the last subset (which usually includes the majority of vertices of the graph) is the only one producing significant interference. However, this is still not effective when the partial orderings are finally merged together. On the contrary, generalized degree produces interference only within the first subset.



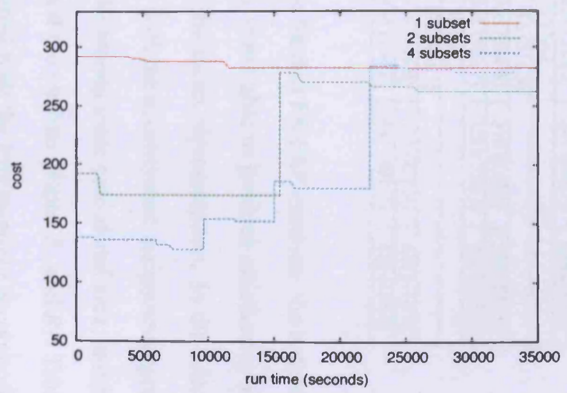
C_1 - generalized degree - 65 freq.



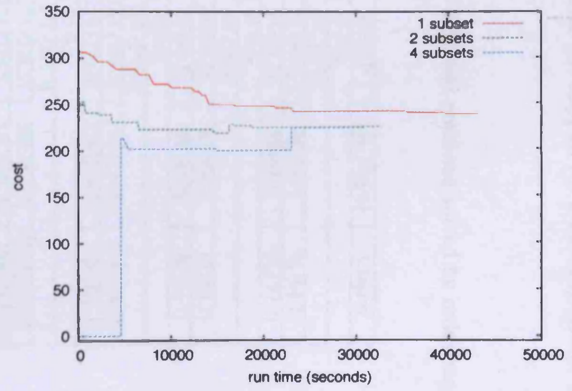
C_2 - GPFAP - 65 freq.



C_2 - clique - 65 freq.



C_3 - GPFAP - 50 freq.



C_4 - GPFAP - 50 freq.

Figure 6.1: Cost-Time plot for the MS/FS-FAP benchmarks with a fixed spectrum of frequencies solved by the order-based GA with decomposition into two and four subsets

Similar behavior is shown by GPFAP (used with the cost in Problem 3.3), which also solves the 'hardest' part of the problem first. However, for the 'two-towns' problem C_4 decomposed into two subsets, GPFAP is still capable of producing a balanced distribution for the interference of the two partial assignments within the two subsets (note that this does not happen for a decomposition in four subsets).

Table 6.1: Best and mean cost for the FS-FAP test problems solved by order-based GA with decomposition (500,000 evaluations)

ns	Random	Geog.	GPFAP	Gen. Degree	Cliques
			C_1 65 freq.		
1			94 (95.7)		
2	144 (144.7)	100 (103.3)	**92 (94.0)**	95 (95.7)	98 (98.7)
4	144 (147.3)	110 (112.7)	94 (95.3)	**92** (94.3)	96 (97.3)
			C_2 65 freq.		
1			49 (50.3)		
2	78 (80.0)	68 (60.7)	40 (42.0)	42 (44.0)	52 (52.3)
4	86 (87.3)	80 (82.0)	48 (48.3)	**39 (41.3)**	50 (51.7)
			C_3 50 freq.		
1			282 (283.0)		
2	358 (359.0)	274 (274.3)	**262 (263.3)**	274 (274.0)	298 (299.0)
4	354 (355.3)	320 (320.7)	278 (279.0)	276 (276.3)	326 (326.7)
			C_4 50 freq.		
1			238 (240.3)		
2	272 (273.0)	227 (228.3)	225 (226.3)	230 (230.7)	240 (241.3)
4	284 (284.7)	229 (229.7)	**224 (225.7)**	244 (245.7)	238 (239.3)
			G_1 30 freq.		
1			316 (316.3)		
2	408 (409.0)	–	**305 (306.3)**	318 (318.7)	320 (320.3)
4	410 (411.3)	–	298 (300.7)	328 (328.3)	312 (313.0)
			G_2 30 freq.		
1			439 (441.3)		
2	526 (527.0)	–	**416 (416.0)**	425 (425.3)	416 (416.7)
4	552 (553.3)	–	436 (436.7)	431 (431.7)	436 (437.0)

As anticipated in Section 6.1, for the hardest FS-FAP instances the order-based representation applied to the GA in 5.1 is not able to produce satisfactory results and, consequently, we need to swap to the direct representation. In the following we will present results of the direct GA with the decomposed assignment approach for the hardest of the FS-FAP benchmarks among those considered here, namely G_2 and G_3, the Philadelphia instance P_2, and the random graphs R_1 and R_2. Table 6.3 compares the results of G_2 and G_3 obtained with the *minimum-cut* decompositions in Definition 6.4 with those obtained with GPFAP and generalized degree.

Table 6.2: C_1 and C_4 with decomposition - subsets solved independently (500,000 evaluations)

	ns	1st sub	2nd sub	3rd sub	4th sub	cost	Table 6.1	global
C_1	GPFAP							94
	2	72	0	-	-	236	92	
	4	30	0	4	0	186	94	
	Generalized-degree							
	2	56	0	-	-	172	95	
	4	24	0	0	0	180	92	
	Clique							
	2	0	48	-	-	158	98	
	4	0	0	0	16	160	96	
C_4	GPFAP							238
	2	156	134	-	-	360	225	
	4	34	0	0	0	355	224	
	Generalized-degree							
	2	166	4	-	-	306	230	
	4	28	0	0	0	366	244	
	Clique							
	2	8	128	-	-	382	240	
	4	8	0	0	138	428	238	

Table 6.3: Best and average cost for test problems G_1 and G_2 solved by the direct GA with decomposition (500,000 evaluations)

	subNo	min-cut1		min-cut2	GPFAP	Gen. degree
		Sequential	Independent			
G_2	1	358 (358.3)				
	2	358 (358.0)	358 (358.0)	349(350.7)	**292 (297.0)**	311 (311.7)
30 freq.	4	–	–	359 (359.3)	282 (284.7)	311 (314.7)
G_3	1	1280 (1280.7)				
	2	1280 (1280.3)	1280 (1280.3)	1275 (1276.7)	**1220 (1223.3)**	1271 (1272.7)
30 freq.	4	–	–	1266 (1272.7)	1255 (1261.0)	1280 (1282.3)

Although the decomposed approach appears to be successful overall, minimum-cut appears to be the weakest of the methods considered. Note that if we consider only the decomposition into graph components, the decomposed technique produces the same cost as the whole approach. Moreover, the 'component' decomposition in Definition 6.4 can be solved either sequentially or independently producing exactly the same costs since no edges are existing between the two components and, as a consequence, none of the constraints is ignored by the decomposed procedure at any stage. However, this decomposition strategy is outperformed by the other decompositions tested, with the GPFAP producing the best outcomes, thus

Table 6.4: Best and mean cost for test problems P_2 solved by the direct GA with decomposition (500,000 evaluations)

	subNo	cell demand		GPFAP
		50% – 50%	80% – 20%	
P_2	1	**203 (206.3)**		
	2	226 (230.0)	210 (211.3)	238 (240.3)
30 freq.	4	232 (236.7)	224 (224.3)	255 (257.3)

confirming the results obtained in Figure 6.1 for the order based GA.

For the Philadelphia problem P_2 the decomposed assignment approach has difficulty in solving FS-FAP instances. This is caused by the very high cell demand of this particular group of benchmarks, in which the final interference produced by the optimal assignment is almost completely dominated by the co-site constraints between transmitters belonging to the same cell. As a consequence, the interference produced by the first (or the first group) of subsets is almost always zero, thus compromising the effectiveness of any decomposition method here used.

Table 6.4 shows the results obtained by the GPFAP decomposition. However, because of the high graph density the decomposed assignment approach appears unsuccessful overall for this specific class of cellular problems when used to solve the FS-FAP.

The random graphs benchmarks already tested for the MS-FAP have been here applied to a limited number of frequencies (150 channels in our experiments). As already mentioned the high graph density of these data sets together with their peculiar structure (which does not present any 'attractor' nodes according to the terminology used for the graph clustering in [41]) does not intuitively make them good candidates for the decomposed assignment approach. This is actually confirmed by the results shown in Table 6.5.

In fact, while for the MS-FAP the decomposed assignment technique was still able to equalise the results produced by the whole approach, the same does not happen for the harder FS-FAP. GPFAP appears overall to be the most competitive

Table 6.5: Best and mean cost for test problems R_1 and R_2 solved by the direct GA with decomposition after 500,000 evaluations

	subNo	GPFAP	Gen. Degree	Cliques
R_1	1	**299 (302.0)**		
	2	304 (304.3)	313 (314.0)	311 (313.0)
150 freq.	4	322 (324.7)	331 (331.7)	327 (327.3)
R_2	1	**203** (204.3)		
	2	215 (217.6)	225 (226.3)	**203 (204.0)**
150 freq.	4	230 (232.0)	255 (255.3)	206 (207.7)

method (at least with a basic decomposition into two subsets). Nevertheless, due to extremely high edge density, the number and the cost of the inter-edges of the partition produced is still very high, thus limiting the effectiveness of the graph partitioning procedure itself. As a results, the first subset (or group of subsets) always produces an interference free assignment whereas a marked discontinuity, which considerably raises the final cost, appears with the introduction of the remaining constraints in the last subset in the sequence (see for example the cost-time plots in Figure 6.2). This tendency is even more important for the other decompositions used.

Note that an exception exists for clique decomposition for the benchmark R_2, in which the decomposed assignment technique matches the best cost found by the whole approach. However, this only appears to be a successful result. In fact, the maximum level zero clique for this specific data sets only includes less than the 2% of the total number of vertices, thus making this approach almost identical to the non-decomposed one. On the contrary, for the second data set considered, in which the size of the maximum level zero clique is slightly more important (about the 5% of the size of the vertices set), the decomposed approach appears unsuccessful when compared with the whole (for clique as well as for the other decompositions tested).

In conclusion, for the FS-FAP the decomposed assignment approach is capable of producing better results than the whole approach for the most of the test prob-

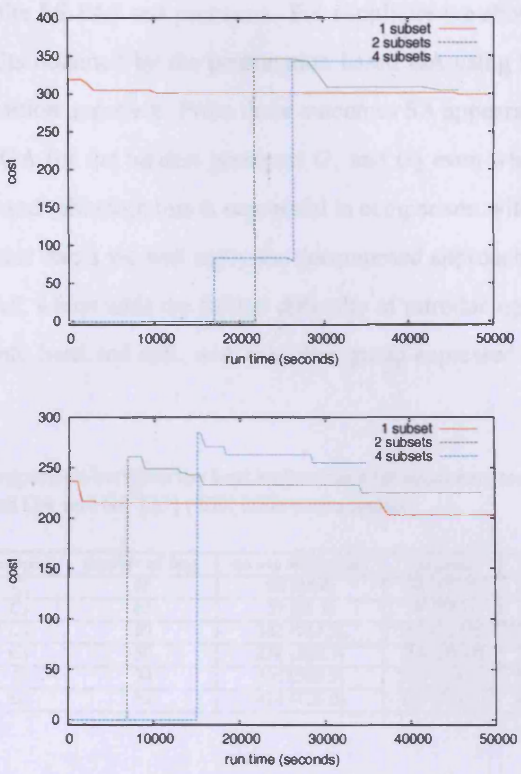


Figure 6.2: Cost-Time plot for the random graph benchmarks with a fixed spectrum of 150 frequencies solved by the direct GA with GPFAP decomposition into two and four subsets

lems considered (with the exception of Philadelphia and random graphs). However, decomposition methods based on minimizing the interconnections between subsets in the partition (i.e. GPFAP) are more effective than those based on isolating and solving first the hardest part of an instance (i.e. clique and generalized degree). This can be partially explained by the fact that for the MS-FAP removing some of the vertices still allows the core part of an instance to produce a span which is close to that of the whole data sets. On the contrary, when the cost is defined in terms of total sum of the constraints violations, as with the FS-FAP, removing edges unavoidably implies a considerable decrease in the partial costs produced by a subproblem.

To summarise Table 6.6 shows a comparison with the best results produced by simulated annealing (with global approach) as previously published in [25] for a selection of the FS-FAP test problems. For simplicity we show only the best and average results obtained by the permutation based GA using both the global and the decomposition approach. From these outcomes SA appears still superior to the order-based GA for the hardest problems G_1 and G_2 even when the decomposed approach is used (although this is successful in comparison with the global). In the next part of this thesis we will apply the decomposed approach to the subcategory of the MI-FAP, which adds the further difficulty of introducing a separation of the constraints into hard and soft, with the latter group expressed in terms of penalty factors.

Table 6.6: Comparison between the best and average (in brackets) costs produced by either the order-based GA and SA [25] (500, 000$*$ evaluations)

benchmark	number of freq.	GA with decomposition	GA global	SA from [25]
C_1	65	**92 (94.0)**	94 (95.7)	127 (127.8)
C_2	65	**39 (41.3)**	49 (50.3)	95 (95.7)
C_3	50	**262 (263.3)**	282 (283.0)	299 (300.4)
C_4	50	**224 (225.7)**	238 (240.3)	252(256.4)
G_1	30	305 (306.3)	316 (316.3)	**291 (293.2)**
G_2	30	416 (416.0)	439 (441.3)	**383 (383.8)**

Chapter 7

Minimum Interference-FAP

Whereas the FS-FAP approach minimizes the maximum interference level expressed in terms of constraint violations, the MI-FAP problem aims to minimize the total sum of weighted interference. As described in Section 1.3.3, hard constraints represent the channel separation required between pairs of transmitters and must be respected in the optimal assignment while soft constraints represent a probabilistic measure of acceptable interference. They are expressed in terms of co-channel and adjacent channel interference.

The hard constraints relate directly to the binary model of the interference (in which one weighted edge connects each transmitters pair) while the soft constraint values reflect a more realistic measure of the interference inside the network. Hence they can be interpreted as an intermediate state between the binary model and more complex ones, such as multiple interference [86].

7.1 Heuristic algorithms

The order-based GA used for the MS-FAP and FS-FAP is not competitive with other meta-heuristics when used to solve the harder MI-FAP instances. This is is true even if cycle crossover is replaced by merge crossover (see [11] and Appendix C.1). Updated results for the COST-259 benchmarks are widely available from the web at [2]. We can note that the latest outcomes are almost all produced by algorithms which include some kind of local optimization procedures (see Section 2.3.2 for more details). However, a standard implementation of simulated annealing obtains results almost as good as those produced by the best performing algorithms [13, 117].

For this reason, SA will become the standard heuristic on which we will focus in order to investigate the effectiveness of the decomposed approach for this more complex model of FAP. When effective the decomposed approach can be seen as a valid alternative to more elaborate algorithms which are more equipped to solve hard instances, for example by adopting complex local optimization techniques.

The idea of decomposing the hardest FAP benchmarks into a partition of subproblems is particularly important for standard meta-heuristics, which are not otherwise capable of producing satisfactory performance within reasonably short periods of times for the more complex types of FAP such as the MI-FAP. On the contrary, the use more complex algorithms on very hard problems may become problematic in terms of computational complexity.

Although its results are worse than those produced by SA, an standard implementation of the GA has still been used for comparison in some of the instances. Here, one of the issues was to explore the potentiality of problem decomposition when used with different types of algorithms. Since the performance of GAs is poor in general on these benchmarks (they do not appear in any implementation in the range of the currently best performing algorithm [2]), it is interesting to investigate if and to what extent their performance can be improved by the decomposition technique.

7.1.1 Simulated Annealing

Simulated annealing has been successfully applied to the different types of FAP considered in this thesis. We have here considered the standard implementation proposed in [60], which is outlined in Algorithm 7.1. From a tuning of the parameters (see Appendix C.2.1) we chose an initial temperature of 0.5 and the basic 'single move', which consists of changing a single frequency value assigned to a selected transmitter.

A second aspect is the choice of the cooling scheme. In our approach we use a number of iterations I equals to the number of transmitters in the current subset and we calculate the reduction index α in order to satisfy the desired total number of evaluations (expressed by a multiple of the total number of transmitters $|V|$). This guarantees that the solutions obtained with both the whole and the decomposed assignment approach consider the same total number of evaluations. It also ensures that the number of evaluations at each temperature is proportional to the current

Algorithm 7.1 SA implementation for the FAP

```
Input:    initialTemperature $t_0$, finalTemperature $t_{min}$, number of iterations $I$,
          reductionIndex $\alpha$, interference Graph $G[V_j]$
Output    FrequencyAssignment $f$

 1: Initialize the temperature $t \leftarrow t_0$
 2: Generate a random assignment $f_{old}$ of the set of transmitters
 3: Evaluate the cost $C_{old}$ of $f_{old}$.
 4: while $t > t_{min}$ do
 5:   for $i = 1$ to $I$ do
 6:     Generate a new configuration $f_{new}$ by changing the frequency of a
        randomly chosen transmitter. Frequencies are chosen at random within
        the transmitter domains.
 7:     Calculate the new cost $C_{new}$
 8:     Calculate $\Delta C = C_{new} - C_{old}$
 9:     if $\Delta C < 0$ or random $<$ prob $= e^{-\frac{C_{new}-C_{old}}{B \cdot t}}$ then
10:       $f_{old} \leftarrow f_{new}$
11:       $C_{old} \leftarrow C_{new}$
12:     end if
13:   end for
14:   reduce $t$ (e.g. $t = \alpha t$)
15: end while
16: Return the final assignment $F \leftarrow f_{old}$
```

subset size. This constitutes a fair basis for comparison.

7.1.2 Genetic Algorithm

Direct Representation

For the MI-FAP instances the order based representation appears incapable of producing satisfactory results. Results can be improved by swapping to the direct representation. With this representation individuals are constituted by a vector whose elements are the frequency values assigned to the corresponding transmitters. This permits complete exploration of the search space.

The framework used is the generational GA NGSAII, which was preferred to SEAMO described in Section 5.1, since it performs better with this type of representation and appears able to produce a better spread of the Pareto non-dominated set [34]. We have used a population size of 20 individuals, as described in Appendix C.2.2. The number of generations is calculated in order to satisfy the desired number of evaluations for a specific run.

The generic procedure for the *Nondominated Sorting Genetic Algorithm II* (NGSA-II) creates at the end of each generation a mating pool by combining the parent and offspring populations. Then each individual is ranked based on its non-dominance (in terms of Pareto dominance) towards the other members of the population. (i.e. the non-dominated set will constitute rank 0 and then it will be removed from the population. Then the next set of non-dominated individuals will form rank 1 and so on). Individuals in the mating pool are then ordered according to their rank values using a fast nondominated sorting approach with computational complexity $O(popSize, M)$ (where $popSize$ is the size of the population and M the number of objectives). Subsequently the selection operator selects the best solutions from the mating pool. This is is based on both the fitness of the individuals (i.e. their non-dominated rank value) and their spread in the objective space, which is obtained by a so called 'crowding distance procedure' (i.e. we will not include in the new generation individuals which are too 'close' in the M dimensional objective space). When a new population of $popSize$ individuals is formed then the genetic operators are applied to it to form a new mating pool (i.e composed by the parents of this generation plus their offspring). Then the whole procedure is repeated by ranking the new mating pool and so on. More explanations and details about the NGSA-II algorithm are given in [34].

Unfortunately, it is not easy to identify in the literature a standard version for this type of GA, with the main difficulty arising from the choice of effective genetic operators. For crossover in particular, the standard versions commonly used for other combinatorial problems appear to be too disruptive for the FAP, thus leading to unsatisfactory results when applied to the MI-FAP. Consequently, authors have proposed specialized operators which are specifically developed for the particular data sets considered. However, none of them has been effectively applied to the COST259 instances tested in this chapter. After some tests (see Appendix C.2) we have adopted the following operators. The crossover used is a variation of those proposed in [31, 68] (whose procedure is outlined in Algorithm 7.2). It has been applied with a probability of 80%.

Algorithm 7.2 Crossover operator for the direct GA

```
1: Find a pair of transmitters $u$ and $v$ for which the constraint between them is
   satisfied, i.e $\varphi_{FS,MI}(f, uv) = 0$ for any of the two currently selected parents
2: If no pair can be found in a fixed number of selections select only one transmitter
   which has no constraint violation in any of the currently selected parents.
3: Interchange the frequencies assigned to u and v and all the vertices belonging to
   their common neighborhood in the first parent with those assigned in the second
   parent.
```

The mutation used, called *swap mutation* (see [68]), consists of a number of simple frequency swaps between pairs of transmitters selected at random, according to a given mutation rate. Mutation is applied at a rate of 0.05% per individual. This means hat fore each individual are performed $0.05 * |V|$ single mutations. To improve the GA performance an *iterative 1-opt LS* procedure (see [40]) has been added after offspring generations to search for local optimality.

Binary tournament selection has been used in the selection process. Tournament selection can be seen as a variation of rank selection [49] and generally involves two stages. In the first phase a group of individual is selected from the population given a certain probability. Then the individual with the highest fitness within the group (also called *pool*) is selected whereas all others are discarded. In the binary tournament selection only two individuals are selected to form the pool.

Finally, the deletion of the duplicates, which is also effectively used in other combinatorial problems [23], contributes in adding variety in the population, thus limiting the *genetic drift* effect (see [37] and Section 3.10). The deletion is here conducted at a 'phenotype' level, which means that individuals will be removed from the population if they are identical in the objective space, as already described in Section 3.10.

Pseudocode for the different phases of this implementation are outlined in Algorithms 7.3 and 7.4.

Algorithm 7.3 direct GA implementation for FAP (NGSAII)

```
Input:  population size $popSize$, number of generations $G$, number of objectives $M$
        interference Graph $G[V_j]$
Output  Non-dominated set $F_1$, FrequencyAssignment $f$

1:  Create a random population $P_0$ of integer vectors representing frequency assignments.
2:  Apply binary tournament selection, crossover and mutation to
    create a child population $Q_0$
3:  while $t \geq 1 \wedge t \leq G$ do
4:      Combine parent and children population $R_t = P_t \cup Q_t$
5:      $F = fastNonDominatedSort(R_t)$ where $F = (F_1, F_2, ..)$
        is the set of all non-dominated fronts of $R_t$
6:      while $|P_{t+1}| < N$ do
7:          Calculate crowding distance in $F_i$
8:          $P_{t+1} = P_{t+1} \cup F_i$
9:      end while
10:     Sort $P_{t+1}$ in descending order using the crowded comparison operator $\geq n$
11:     Select the first $N$ elements of $P_{t+1} = P_{t+1}[0 : N]$
12:     while $|Q_{t+1}| < N$ do
13:         Use selection, crossover and mutation to generate a new individual $q =$
            $makeNewPop(P_{t+1})$
14:         if q is not a phenotype duplicate in the population then
15:             Add q to the new population $Q_{t+1} = Q_{t+1} \cup q$
16:         else
17:             It dies.
18:         end if
19:     end while
20:     Apply LS 1opt to $Q_{t+1}$ to search for local optimality
21:     t=t+1
22: end while
23: Return the final non-dominated set $F_1$
24: Return the individual $f$ with the best global cost $\sum_{m=1}^{M} obj_m$
```

Order-based representation

For the reasons mentioned in the previous section the order-based GA will not be actually used for the MI-FAP. However, a small number of experiments will be still presented for comparison between the two representations used in this thesis for the GA.

The order-based representation will be now used within a NGSA-II framework

Algorithm 7.4 Iterative 1-OPT implementation for the FAP

```
Input:   FrequencyAssignment $f_{old}$
Output   FrequencyAssignment $f$
1: Select a random permutation ord of the transmitters
2: while no more cost improvements do
3:    while next transmitter in ord do
4:       Reassign transmitters sequentially to the best frequency
         in the domain according to the ordering ord.
5:       Update assignment $f_{old}$
6:    end while
7:    Select a new random permutation ord of the transmitters
8: end while
9: Return the final assignment $f \leftarrow f_{old}$
```

adopting the same parameters for the population size and number of generations used for the direct GA described above. A newly proposed crossover called *merge crossover* (see Appendix C.2.2), which has been effectively used in the other similar problems such as *graph coloring*, has been applied with a probability of 100% [11]. For a description of the method see Appendix C.2.2. One order based mutation per offspring will be applied (as in the experiments conducted for the MS/FS-FAP with this representation). In addition, the same 1-opt LS already introduced for the direct GA will be also used for this representation, see also Section 5.1. Details about the tuning tests performed with this GA are given in Appendix C.2.2.

7.1.3 Multi objective approach

The NGSA-II framework implemented for the MI-FAP presents another important difference from the other versions of the GA previously introduced. We have introduced a novel *multi-objective approach* which can be seen as alternative to the introduction of penalty factors mentioned in Section 2 (a technique which presents difficulty in finding a suitable setting for the weights).

If we solve the GA as one objective optimization, as it is normally suggested in order to solve the FAP, we minimize the total interference which, in the case of the MI-FAP, is only composed by the sum of the two types of soft constraints (assuming that we reach a nearly optimal solution which satisfy all the hard constraints).

The idea is now to solve the same problem as a two objective optimization problem in which the two different objectives are constituted by the two different types of soft constraint, respectively *co-channel* and *adjacent-channel* interference.

Definition 7.1 *Give an assignment let be E^{coch} the edges corresponding to a violation of the co-channel constraint (i.e. their end points are assigned the same frequency) and E^{adj} the edges corresponding to a violation of the adjacent-channel constraint (i.e. their end points are assigned with one channel separation). The solving a MI-FAP problem as a one objective optimization problem (with the notation used in Problem 1.3) consist of minimizing:*

$$O_{MI}(f) = \sum_{uv \in E^{coch}} \varphi_{MI}(f, uv) + \sum_{uv \in E^{coch}} \varphi_{MI}(f, uv) = O^{coch}_{MI}(f) + O_{MI}adj(f)$$

If we solve the problem as a two objective optimization the two objective considered are then $O^{coch}_{MI}(f)$ and $O^{adl}_{MI}(f)$.

The two objective implementation maintains a population of individuals which do not dominate each other as the two objective were competitive and cannot being optimized as they would be independent, i.e. the optimization of one objective cannot be done without penalizing the other objective. This in our case is introduced as an artificial expedient to preserve diversity while at the end of the run the algorithm returns the solution with the best (minimum) total cost $O_{MI}(f)$ within all of the individuals in the final e population (as in the usual one objective optimization problem). Note that the NGSAII framework described in 7.3 has been expressively designed for multi-objective optimization and it is based on the concept of Pareto dominance and the maintenance of the non-dominated fronts, which are approximations of the Pareto optimal front, see [33].

The multi-objective choice presents the advantage of reducing considerably the problem of *premature convergence* (see Section 3.10) which is one of the main problems of the single objective approach. Therefore, this can be seen as an alternative to the choice of introducing other artificial expedients, such fitness sharing used in Section 6.1. Moreover, it can provide information about which type of interference may be predominant (i.e. it can identify the nature of the trade off between co and adjacent channel interference).

Tables 7.1 and 7.2 show a comparison between the one and the two-objective approaches for the Siemens1 and the Siemens2 benchmarks respectively. In both cases the direct GA has been run for 2,000 generations with a population of 20 individuals ($1,000,000 * |V|$ total evaluations approximatively).

Table 7.1: Best and average cost O_{MI} (over three runs) for the COST259 Siemens1 benchmark solved by the one and two-objective direct GA

subsetsNo	one obj.	two obj.
1	4.14 (4.36)	4.02 (4.13)
2	3.79 (3.84)	3.53 (3.57)
4	4.05 (4.07)	3.46 (3.59)

Table 7.2: Best and average cost O_{MI} (over three runs) for the COST259 Siemens2 benchmark solved by the one and two-objective direct GA

subsetsNo	one obj.	two obj.
1	18.45 (18.72)	18.00 (18.14)
2	19.27 (19.38)	18.88 (19.03)
4	21.62 (21.02)	21.50 (21.79)

We can firstly observe that the figures in the table show only a little improvement in the results produced by the two-objective choice. This is can be partially caused by the fact that the 1-opt local optimization procedure is applied to the global objective $O_{MI}(f)$, composed of the sum of the co-channel and adjacent-channel, see Definition 7.1. This tends intuitively to uniform the one objective with the two objective minimization approach. Alternatively, the two objectives can be weighted at random, in order to induce a further spread of the Pareto set of the non-dominated solutions, with the distribution of weights changing, for instance, at each generation. The total composed weight will now depend on the pair of weights (W_1, W_2) used as:

$$O_{MI}(f, W_1, W_2) = W_1 \cdot O_{MI}^{coch}(f) + W_2 \cdot O_{MI}^{adj}(f)$$

Few more experiments were conducted with this further expedient without, however, producing significant improvements, thus their results will be omitted from the rest of the thesis.

Nevertheless, in our experiments, the two objective approach is more successful in delaying the convergence of the algorithm and in reducing the negative phenomenon of genetic drift. In some of the tests the one objective approach outperforms the two objectives one during the first part of the run but its result do not further improve after a small number of generations (see for example Figure 7.1 which shows a single run of Siemens1). On the contrary, the approximations of the non-dominated set of the partial solutions in the two objectives space continues, even if rather slowly, to advance towards the final approximate Pareto set. Figure 7.2 shows an example of the behaviour of the approximated non-dominated set through the generations.

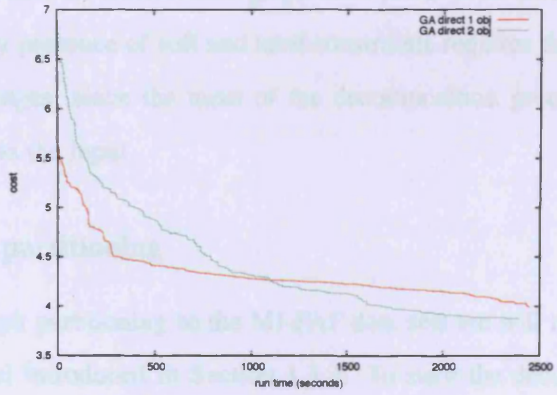


Figure 7.1: Cost-time plot for Siemens1 solved by direct GA with one and two objectives after $1,000,000 * |V|$ evaluations

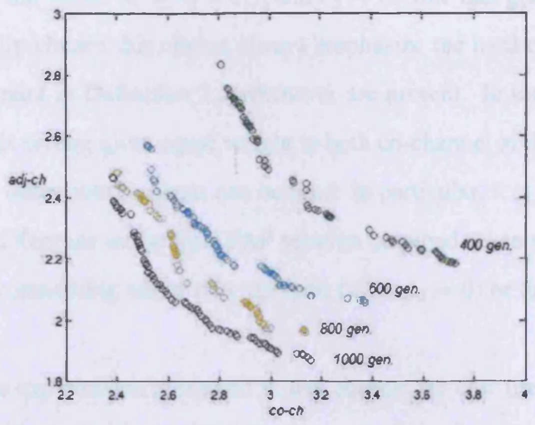


Figure 7.2: Approximated non-dominated set through the generations during a run of the COST259 Siemens1 benchmark

7.2 Decomposition algorithms

This section describes the decomposition algorithms tested for the MI-FAP. Although they will mainly be those already described in the previous section, some modifications are needed to adapt them to the specific formulation of this type of FAP.

Before presenting the modified versions of the decomposition methods it is important to clarify how the interference graph itself needs to be modified too. In fact, the contemporary presence of soft and hard constraints requires the introduction of a single set of edges, since the most of the decomposition procedures rely on a weighted graph as the input.

7.2.1 Graph partitioning

To adapt the graph partitioning to the MI-FAP data sets we will refer to the graph theoretical model introduced in Section 1.3.3. To state the decomposition algorithms used at step 1 of Algorithm 3.1, we first define a weighted simple graph G^D that combines the hard and soft constraints.

Although most of the results shown in this section are obtained by setting these parameters to the values $\lambda_1 = \lambda_2 = 0.5$, and $\lambda_3 = 1$. Note that given c_{uv}^{hard} as a high value arbitrarily chosen this choice always emphasize the hard constraints, which become dominant in Definition 3.2 whenever are present. In the absence of hard constraints this setting gives equal weight to both co-channel and adjacent channel interference), other combinations can be used. In particular, it is interesting to test the eventual difference in the final FAP solution obtained when the partitioning is produced by considering either only the hard ($\lambda_1 = \lambda_2 = 0$) or the soft constraints ($\lambda_3 = 0$).

For all the experiments presented in this chapter the cost used in Problem 3.4 has been used for the balanced GPFAP whereas for the unbalanced GPFAP the cost formulated in Problem 3.5 has been preferred. Both of these costs have been chosen because they produced better results in a number of test problems performed (see also [26]). In particular the latter cost takes advantage of the graph clustering parameters used, that is the inter and intra clustering conductance, in order to maintain a more balanced partitioning, thus avoiding trivial decompositions that includes all transmitters in one subset only.

To solve both types of GPFAP we used the memetic GA proposed in Algorithm 3.10. The number of subsets and the remaining parameters have been set according to the values used in [26] in some preliminary test problems. The population consists of 100 individuals and the algorithm is run for 500 generations. Cycle crossover is applied with a rate of 100% whereas one single order-based mutation is performed for each of the offspring generated. Finally fitness sharing is used as a niching method with the same implementation used in [23] and outlined in equation (3.6). The LS added in order to speed up the process was a SA run for 1000 iterations for each of the offspring produced. Figures 7.3 visualizes the results of the graph partitioning decomposition into two subsets for Siemens2 (balanced and unbalanced GPFAP) while Figure 7.5 shows the two towns test problem C_6 (balanced GPFAP). Note that for the latter instance the plot shows how the transmitters belonging to different towns are largely separated into different subsets.

7.2.2 Graph clustering

The Markov graph clustering with the standard setting of the parameters (see [120]) tends to create a large number of clusters which are then ineffective in solving FAP instances. In fact, any decomposition method produces in general poor results when the number of subsets in the partition becomes too high (usually greater than four or five). However, the setting of the parameters cannot be easily changed without affecting the computational complexity of the algorithm. As a consequence, the Markov algorithm produces a number of clusters (usually in the order of tens) which can still be higher than needed for our experiments.

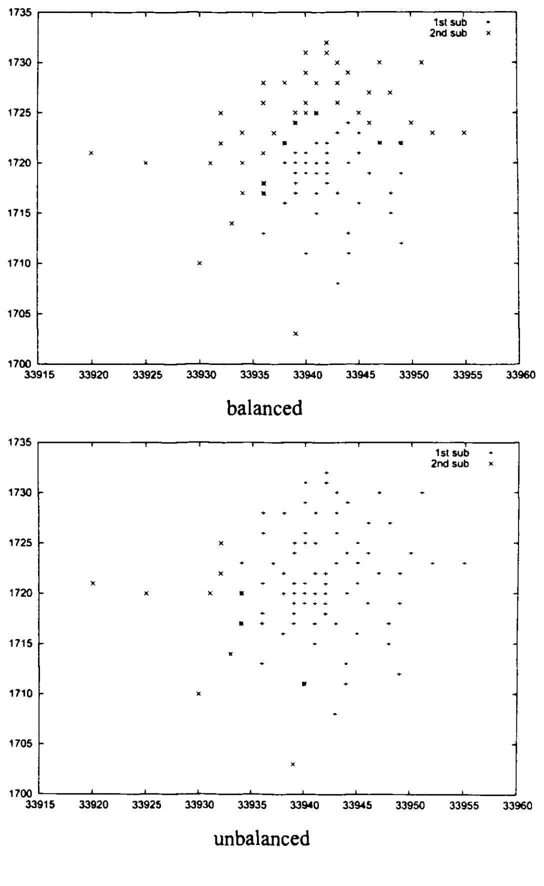
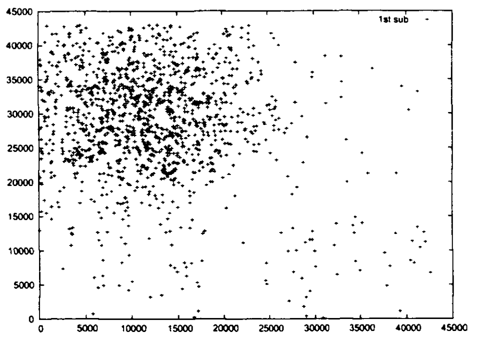
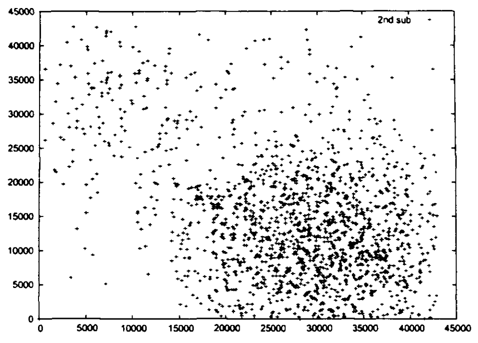


Figure 7.3: GPFAP decomposition into two subsets of the Siemens2 COST-259 data set



balanced



unbalanced

Figure 7.4: Balanced GPFAP decomposition into two subsets of data set C_6

We will refer to the resulting partitioning as a *Markov clustering*. To reduce the clusters to the desired number, usually two to four, we have adopted the procedure in Algorithm 7.5 in order to merge selected clusters. Note that the resulting partitions will not be necessarily balanced since the subsets constituting the Markov clustering $C(G)$ usually contain a variable number of vertices of the original graph G. For this reason, its performance is expected to be similar to the other methods which produce an unbalanced partitioning, such as unbalanced GPFAP and clique.

7.2.3 Other decompositions

Generalized degree decomposition is simply obtained by applying Algorithm 3.6 to the graph G^D. Further modifications of the graph need to be introduced to deter-

Algorithm 7.5 Merge selected clusters

```
Input:  Partition $M(G^D) = \{ M_1, M_2, \dots, M_m \}$, $noSubs$
Output: Partition $V(G^D) = \{ V_1, V_2, \dots, V_{noSubs} \}$
1: Given a Markov clustering $M(G^D)$ in $m$ subsets $\{ M_1, M_2, \dots, M_m \}$ of the graph
   $G^D(V, E)$ we consider an artificial graph $G^c(V, E)$ in which:
     • vertices $v_i \in V(G^c)$ represent the individual clusters $M_i$ of the partition
       $M(G^D)$.
     • edges $ij \in E(G^c)$ between pairs of different subsets are assigned weight
       values equal to the sum of the inter-edges of $G^D$ which belong to subsets
       $M_i$ and $M_j$:
       $c_{ij} = \sum_{uv \in E_i^{inter}(G^D) \cap E_j^{inter}(G^D)} c_{uv}$
2: Execute a (balanced) graph partitioning algorithm of the graph $G^c$ (see Problem
   3.4) in order to obtain a partition $C(G^c) = \{ C_1, C_2, \dots, C_{noSubs} \}$ into the desired
   number of $noSubs$ subsets.
3: Expand each of the vertices $v_i \in V(G^c)$ by including all the transmitters contained
   in the clusters $M_i$ of $M(G^D)$ to obtain a partition $V(G^D)$ of $G^D$ into $noSubs$
   subsets.
```

mine the clique decomposition. Because of the different weights used for the hard and soft constraints (which reflect the different importance of these two different categories), the adoption of the weighted procedure in Algorithm 3.7 to detect the maximum clique of G^D appears more suitable. Note that to apply this procedure we need to modify the graph G^D as described in Definition 3.3 to obtain a suitable graph G^C. Finally, in order to determine partitions with more than two subsets the same procedure has been applied recursively, as previously proposed in Algorithm 3.8 for the MS\FS-FAP.

Geographical decomposition will be applied for comparison in the few benchmarks provided with location information (that is the Siemens and Cardiff instances). Figure 7.5 shows the results of the geographical decomposition into two subsets for Siemens2 obtained by applying Algorithm 3.4. Random decomposition will be not considered for this harder type of FAP. Because of the presence of the hard constraints a decomposition drawn at random is very likely to produce final

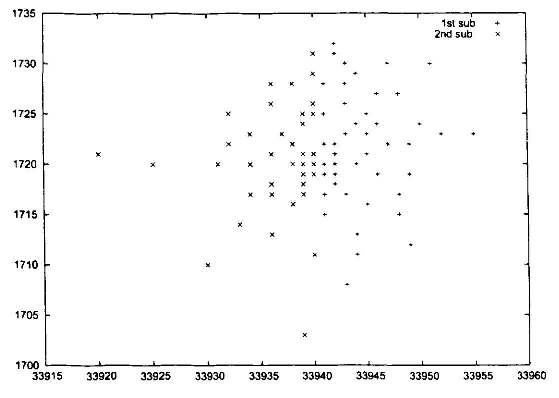


Figure 7.5: Geographical decomposition into two subsets of the Siemens2 COST-259 data set

assignments which are not feasible for this problem (i.e. which are violating some of the hard constraints).

The different importance between hard and soft constraints in the interference graph plays a primary role in the effectiveness of a decomposition. As described in Section 1.3.1 the former category of constraints must be satisfied in a feasible assignment and, for this reason, is associated with very high artificial weight values whereas the latter is normally weighted in the range of few units and constitutes the actual objective of optimization, see Table 4.7. Hence, a decomposition cannot disregard the distribution of the hard constraints. Whenever many of their corresponding edges are included in one of the inter-edge sets between distinct pairs of subsets, is more likely that the resulting solutions of the FAP will not be feasible. Figures 7.6 shows the distributions of the hard constraints for the Siemens instances. Note that since transmitters within a cell have all the same geographical coordinates it is not possible to represent intra-site constraints and so the constraints shown are only handover constraints. We can observe how some of them, e.g. Siemens2, are more difficult to be partitioned into subsets whereas others show a more natural separation into clusters, see for instance Siemens3 which also presents a small disconnected component.

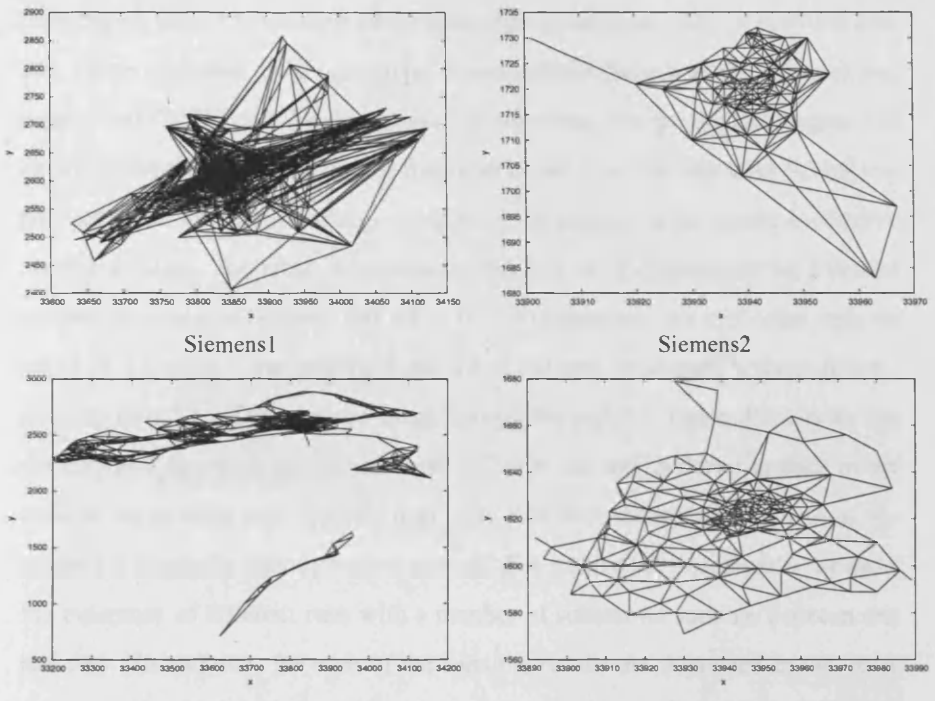


Figure 7.6: Hard constraints distributions for the COST259 Siemens instances

7.3 Experimental results

In this section we will present the results obtained by applying the decomposed assignment approach proposed in Algorithm 3.1 to the MI-FAP benchmarks described in 4.2.2. We will firstly consider a number of runs of the COST259 instances in order to compare the performance of different decomposition techniques. Subsequently, we will presents the outcomes of longer runs on the same benchmarks together with the Cardiff University ones for the most effective decomposition algorithms. Finally, we will have a closer look at the trade-off between quality and runtime and at the distribution of the local interference when the decomposed approach is applied.

7.3.1 Comparison of decomposition algorithms

Information about the location of the transmitters is available for the Siemens data sets, hence we tested the geographical decomposition along with the balanced and unbalanced GPFAP, the Markov clustering algorithm, the generalized degree and clique decomposition. The aim of the experiments is to exclude from further experimentation any decomposition method which appears to be clearly ineffective for this problem. Therefore, we conducted this first set of experiments for a limited number of total evaluations ($100,000 * |V|$). Furthermore, we will focus only on standard SA since it outperformed the GA in the tests conducted without decomposition (see 7.1). The heuristic loops through the partition twice. Results for the decomposed approach are summarized in Table 7.4, which shows in each of the column the average cost obtained over three runs with different random seeds obtained for a specific decomposition method. For each of the benchmarks we show the outcomes of different runs with a number of subsets ns variable between one and four. In addition, for each of the runs the results are organized in two rows corresponding to the first and second loop through the subsets. Finally, Table 7.3 shows for the same benchmarks the costs obtained with the global approach without any decomposition applied. Results are presented in columns corresponding to a different test problems and are still organized in two rows corresponding to two different loops to maintain consistency for a comparison with those of the decomposed approach. However, in the case of only one subset the results for the second loop actually refer to a second run with double number of evaluations with respect to those obtained with a single loop. The best outcomes are highlighted in bold whereas if the algorithm is unable to produce feasible solutions (i.e. solutions satisfying the hard constrains) the corresponding costs are displayed in blue. However, these do not have any numerical significance since the value produced by the algorithm is in these cases that of the weights c_{uv}^{hard}, which is artificially set to $1,000$ in order to represent the hard constraints.

The GPFAP and the Markov clustering appear superior to clique and generalized-

Table 7.3: Siemens1-4 for SA without decomposition ($100,000 * |V|$ evaluations per loop)

first loop			
second loop			
SIEMENS1	SIEMENS2	SIEMENS3	SIEMENS4
3.38 (3.44)	17.12 (17.67)	8.24 (8.52)	92.43 (92.71)
3.30 (3.37)	**16.75 (16.89)**	**8.14 (8.31)**	**92.12 (92.23)**

degree decompositions, which for the hardest instances have difficulty in producing valid solutions (that is they may contain violated constraints). However, neither GPFAP nor Markov appears to clearly outperform the other in all of the instances tested. Furthermore, there is not a clear difference between the performance of the balanced and unbalanced version. Note that in some instances of the unbalanced version one of the subsets includes the majority of the transmitters, making its solution very close to that of the problem as a whole. This can bring about some advantages in the optimality but can also affect the runtime of the decomposed approach.

Geographical decomposition takes advantage because some of the Siemens instances are decomposed into natural clusters, see for example Siemens3 which also presents a distinct component. However, this method is penalised by the fact that it does not take account of the number and weights of the inter-edges between different subsets.

Clique based decomposition only partially produces a competitive performance. This may be caused by the fact that, because of the small clique sizes, this method often finds a trivial interference free solution for the first subset. As a consequence fixing their assignment has the effect of restricting the assignments of subsequent subsets.

Finally, generalized degree decomposition neither takes into account the inter-edges between subsets or resembles any sort of clustering or unbalanced partitioning. As a consequence it produces the worst performance among all the decomposition methods tested.

Note the effect of the second loop and how this is beneficial both in terms of mean and variance. In physical terms this is related to a locally different interference distribution in the subsets as it will be discussed in more detail at the end of this chapter.

For Siemens1, 3, and 4 the GPFAP technique and the Markov clustering both perform better than solving the original problem as a whole. However, for Siemen2 the decomposition in subsets does not improve the results obtained with only one subset. This will be confirmed by the longer runs presented in the next section. Note that, the results for this benchmarks are better for the unbalanced GPFAP rather than the balanced version. This performance improvement can be explained by the fact that the unbalanced decomposition naturally reduces the number of the inter-edges, making the number of transmitters in at least one of the subsets equal to a very low percentage of the total.

Table 7.4: Siemens1-4 for SA with decomposition ($100,000 * |V|$ evaluations per loop)

ns	Bal. GPFAP	Unbal. GPFAP	Gen. Degree	Cliques	Geog	Markov	Global
	first loop						
	second loop						
SIEMENS 1							
2	3.16 (3.28)	3.38 (3.44)	5.46 (5.72)	4.57 (4.64)	3.58 (3.62)	3.32 (3.33)	3.30 (3.37)
	3.14 (3.18)	3.30 (3.37)	4.24 (4.31)	3.49 (3.58)	3.43 (3.54)	3.26 (3.29)	
3	3.36 (3.45)	3.35 (3.43)	5.89 (6.23)	3.31 (3.38)	3.81 (4.01)	3.40 (3.61)	
	3.32 (3.36)	3.29 (3.34)	4.99 (5.05)	3.42 (3.52)	3.48 (3.61)	3.29 (3.41)	
4	3.15 (3.19)	3.99 (4.20)	6.85 (6.99)	3.59 (3.67)	3.65 (3.72)	3.34 (3.42)	
	3.52 (3.53)	3.30 (3.44)	5.35 (5.56)	3.58 (3.68)	3.45 (3.56)	3.23 (3.27)	
SIEMENS 2							
2	17.91 (17.98)	17.93 (18.10)	23.30 (24.37)	18.38 (20.02)	19.33 (19.54)	21.49 (21.81)	16.75 (16.89)
	17.83 (17.91)	17.59 (17.86)	18.64 (19.04)	18.22 (18.42)	17.96 (18.14)	19.22 (19.34)	
3	19.91 (20.23)	19.79 (19.94)	26.14 (26.26)	19.59 (2,018)	20.47 (20.38)	24.09 (24.82)	
	18.30 (18.57)	17.99 (18.09)	21.26 (21.36)	19.35 (19.44)	18.20 (18.71)	20.18 (20.68)	
4	20.99 (21.10)	17.72 (17.93)	27.47 (27.52)	19.82 (2,686)	20.79 (2,687)	22.91 (23.36)	
	18.60 (18.87)	**17.31 (17.42)**	22.61 (22.76)	19.18 (19.33)	18.95 (19.09)	20.45 (20.80)	
SIEMENS 3							
2	7.66 (7.84)	7.59 (7.69)	2,009 (3,350)	9.14 (674.8)	9.06 (9.24)	8.04 (8.17)	8.14(8.31)
	7.58 (7.81)	7.31 (7.49)	9.15 (1,342)	**8.76 (8.82)**	8.83 (8.95)	7.77 (7.89)	
3	7.30 (7.45)	7.82 (7.99)	2,016 (3,646)	8.48 (675.3)	8.49 (8.81)	9.17 (9.22)	
	7.24 (7.40)	7.79 (7.98)	11.30 (678.0)	**8.03 (8.27)**	8.42 (8.44)	7.54 (7.65)	
4	8.09 (8.28)	8.43 (8.57)	3,017 (3,684)	8.61 (675.6)	9.09 (9.34)	7.55 (7.64)	
	7.98 (8.14)	7.92 (8.13)	12.40 (1,345)	**8.43 (8.48)**	8.40 (8.68)	7.42 (7.44)	
SIEMENS 4							
2	90.65 (91.50)	90.62 (90.88)	2,147 (3,145)	93.04 (723.3)	96.33 (2,761)	92.22 (92.51)	92.12 (92.23)
	90.65 (91.48)	89.54 (90.09)	2,120 (3,082)	**92.62 (93.75)**	92.59 (92.75)	90.10 (90.19)	
3	94.13 (94.85)	88.63 (89.12)	3,553 (3,823)	2,254 (2,371)	96,201 (2,096)	94.04 (94.17)	
	92.88 (93.56)	**88.23 (88.43)**	3,131 (3,514)	**93.92 (95.54)**	93.52 (427.5)	91.43 (91.74)	
4	94.72 (95.68)	92.40 (93.63)	3,749 (3,983)	2,651 (3,001)	3,399 (4,099)	1,215 (2,112)	
	93.19 (93.38)	92.07 (93.42)	3,535 (3,801)	96.39 (97.46)	95.24 (96.25)	94.25 (94.38)	

Table 7.5 shows the outcomes of the same decomposition methods when a reverse ordering is applied, that is the subsets are exactly the same of those used in Table 7.4 but the order of the assignment is reversed. The performances of the GP-

FAP and the Markov clustering are in general worse than in Table 7.4 for all the four Siemens instances. This can be explained by the fact that these procedures lose the advantage of having the first subset which approximatively represents the hardest part of the problem. An opposite tendency is shown by the clique decomposition, which generally improves the results obtained in Table 7.4 instead. However, with the reverse order this decomposition method is actually an unbalanced partitioning with the larger subsets considered at first, which may explain the improvements in its performance. Generalized degree decomposition and geographical decomposition do not seem to be competitive with any of the other methods tested.

Table 7.5: Siemens1-4 for SA with decomposition - reverse ordering ($100,000 * |V|$ evaluations per loop)

ns	Bal. GPFAP	Unbal. GPFAP	Gen. Degree	Cliques	Geog	Markov	Global
	first loop						
	second loop						
				SIEMENS 1			
2	4.07 (4.13)	3.78 (3.95)	22.93 (22.94)	4.57 (4.76)	3.75 (3.85)	3.41 (3.45)	3.30 (3.37)
	4.04 (4.10)	3.78 (**3.94**)	19.42 (19.75)	3.49 (3.73)	3.55 (3.59)	3.39 (3.44)	
3	4.43 (4.48)	3.42 (4.43)	6,025 (8,026)	4.12 (4.43)	3.87 (4.02)	3.45 (3.33)	
	4.06 (4.31)	**3.25** (4.06)	21.81 (22.03)	4.01 (4.06)	3.62 (3.66)	3.29 (3.32)	
4	4.98 (5.02)	4.35 (6.37)	14,025 (14,026)	4.35 (4.71)	3.72 (3.80)	3.75 (3.88)	
	4.30 (4.48)	4.26 (6.93)	22.40 (2,022)	4.19 (4.48)	3.60 (3.73)	3.47 (3.53)	
				SIEMENS 2			
2	19.45 (21.68)	18.24 (18.62)	22.93 (2,022)	4,023 (4,024)	18.88 (19.08)	2,022 (4,021)	16.75 (16.89)
	18.58 (19.88)	18.95 (19.58)	19.42 (19.54)	**17.86 (17.96)**	17.97 (18.24)	18.95 (19.46)	
3	21.69 (23.63)	24.66 (1,358)	4,026 (6,025)	2,026 (4,692)	20.25 (20.58)	6,012 (6,023)	
	18.54 (20.97)	21.32 (21.89)	21.70 (22.03)	18.56 (18.74)	18.57 (18.66)	20.23 (20.61)	
4	21.29 (22.83)	25.93 (3,359)	14,025 (14,027)	6,026 (6,693)	22.38 (689.1)	2,025 (4,025)	
	19.92 (21.02)	21.72 (22.27)	22.09 (688.9)	19.82 (686.6)	19.29 (19.52)	20.70 (20.91)	
				SIEMENS 3			
2	8.35 (11.09)	8.35 (8.63)	3,800 (4,534)	4,010 (9,342)	8.76 (8.94)	9.68 (9.85)	8.14 (8.31)
	7.98 (10.216)	**7.98 (8.15)**	8.52 (1,341)	**7.98** (8.12)	8.59 (8.64)	8.36 (8.59)	
3	10.91 (10.93)	2,010 (2,677)	3,201 (3,736)	4,013 (6,013)	8.26 (8.31)	9.92 (10.25)	
	8.70 (9.10)	**8.52 (9.44)**	2,010 (2,924)	8.38 (8.48)	8.00 (8.23)	8.27 (8.52)	
4	12.71 (12.86)	10.93 (2,013)	6,401 (7,001)	8,013 (10,680)	8.20 (675.0)	9.42 (9.46)	
	10.36 (11.08)	9.70 (10.51)	11.53 (4,010)	8.74 (8.92)	8.20 (8.35)	7.82 (8.21)	
				SIEMENS 4			
2	99.14 (877.5)	97.50 (1,431)	4,093 (6,095)	3,394 (4,091)	98.63 (2,095)	2,092 (4,012)	92.12 (92.23)
	93.32 (197.7)	94.33 (95.14)	2,101 (3,769)	94.06 (94.76)	94.75 (95.91)	94.24 (94.50)	
3	102.13 (6,880)	2,097 (3,084)	7,611 (8,677)	2,710 (6,095)	6,098 (8,095)	6,058 (8,112)	
	94.43 (3,532)	90.76 (90.91)	1,210 (1,610)	93.20 (94.17)	93.74 (94.45)	4,110 (6,015)	
4	101.88 (769.2)	97.48 (97.69)	10,212 (11,287)	14,096 (18,090)	2,098 (6,098)	4,025 (6,115)	
	94.97 (96.80)	94.19 (95.46)	2,211 (3,078)	93.30 (94.88)	95.49 (96.61)	2,084 (4,115)	

To summarise, GPFAP and Markov clustering appear to be the most effective decomposition algorithms for this type of FAP. Furthermore, the use of a reverse ordering does not improve the results produced with the original one used in Table 7.4.

Table 7.6 shows the results obtained for Siemens1 and Siemens2 for a number of runs in which the subsets have been run independently according to Algorithm

3.2. We present the best final cost over three runs together with the corresponding partial costs produced by each single subset and obtained by recomposing the partial assignments to produce a final complete assignment for the whole transmitters in the network. Results are shown for balanced and unbalanced GPFAP for $100,000 * |V|$ evaluations. The costs in Problems 3.4 and 3.4 have been used for the balanced and unbalanced partitioning respectively. From the partial figures in the different subsets it is confirmed that the first subset always produces the highest interference values. Note that for the unbalanced GPFAP the partitioning is re-ordered with the larger subset in the first position. As a consequence, it tends to produce interference free assignment within the last subsets, often composed of only few vertices.

However, the final costs always present infeasible solutions and similar results were obtained for the other decompositions tested. Consequently, this approach can only be used if followed by further local optimization procedures. For example, it can be used as a pre-processing before the application of a generic meta-heuristic procedure, which could include the 'sequential' decomposition technique proposed in Algorithm 3.1 (as will shown as example at the end of the chapter).

Table 7.6: Siemens1 and 2 for SA with decomposition - subsets solved independently (100,000*—V— evaluations per loop)

* no valid solutions

ns	Bal. GPFAP					Unbal. GPFAP					from Alg. 3.1	Global
	1st sub	2nd sub	3rd sub	4th sub	cost	1st sub	2nd sub	3rd sub	4th sub	cost		
SIEMENS 1												
2	1.953	1.171	-	-	26,004	2.253	1.067	-	-	2,003.72	**3.14**	3.30
3	1.008	0.049	0.867	-	204,017	1.128	0.001	0.001	-	446,038		
4	1.205	0.5408	0.069	0.219	460,010	1.227	0.001	0.000	0.000	452,042		
SIEMENS 2												
2	5.050	1.494	-	-	82,027	7.212	0.927	-	-	58,024	17.31	**16.75**
3	1.187	1.142	0.013	-	198,037	1.240	0.176	0.017	-	756,042		
4	0.650	0.292	0.252	0.008	176,041	1.017	0.018	0.000	0.000	900,044		

7.3.2 GPFAP results

A number of longer runs were performed with the GPFAP partitioning for both the balanced and unbalanced versions (see the formulations in Problems 3.4 and

3.5 respectively). This decomposition algorithm produced the best outcomes in the tests discussed in Section 7.3.1, as well as the Markov clustering, but it has a much easier and inexpensive implementation. Moreover, to quickly produce good approximated decompositions for the FAP we have adopted the heuristic procedure in Algorithm 3.10.

Siemens

Firstly, we present the results for the Siemens benchmarks for a higher number of subsets and total final evaluations. Each of the benchmarks was tested for runs of different lengths, from $10,000 * |V|$ to $2,000,000 * |V|$ total evaluations, in order to analyze differences in the behavior between the solutions obtained with and without decomposition.

We will present in this Section only the results for the longest run while we refer to Appendix A.3 for the complete results. Tables 7.7, 7.8, and 7.9 show the the best and average cost over three runs obtained by SA and the GA with and without the decomposed assignment approach.

Table 7.7: Siemens1-4 for SA without decomposition ($2,000,000 * |V|$ evaluations per loop)

first loop				
second loop				
	SIEMENS1	SIEMENS2	SIEMENS3	SIEMENS4
SA	2.75 (2.83)	15.72 (15.79)	6.61 (6.72)	87.25 (87.98)
	2.68 (2.76)	**15.59 (15.64)**	**6.59 (6.62)**	**86.59 (87.12)**
GA	3.96 (4.01)	18.00 (18.14)	9.77 (10.20)	105.69 (106.42)
	3.61 (3.72)	17.64 (17.91)	9.54 (9.77)	101.05 (102.14)

For both the GA and the standard SA, the decomposed assignment approach produces better results than the problem solved as a whole in three out of four of the benchmarks tested, as in Table 7.4. Histograms in Figure 7.7 show the behaviour of cost against number of evaluations for average runs of the Siemens benchmarks with and without the decomposed assignment approach. Note that the longer the runs, the less is the percentage improvement produced by the decom-

Table 7.8: SA - Siemens1-4 with GPFAP decomposition ($2,000,000 * |V|$ evaluations per loop)

† at least 1 invalid solution * no valid solutions

	noSub.	SIEMENS1	SIEMENS2	SIEMENS3	SIEMENS4
		first loop			
		second loop			
Bal. GPFAP	2	2.68 (2.75)	16.97 (17.04)	6.39 (6.46)	84.35 (84.80)
		2.61 (2.66)	**16.34 (16.73)**	**6.37 (6.44)**	**84.08 (84.39)**
	3	2.95 (3.01)	19.54 (19.77)	6.53 (6.81)	89.50 (90.59)
		2.93 (2.95)	17.41 (17.60)	6.46 (6.58)	87.16 (88.51)
	4	2.94 (3.01)	20.56 (20.87)	6.95 (7.02)	89.96 (90.43)
		2.90(2.98)	18.02 (18.27)	6.79 (6.89)	89.53 (90.17)
	5	3.34 (3.43)	20.31 (20.46)	7.65 (7.79)	92.94 (93.5)
		3.04(3.13)	17.90 (18.53)	7.26 7.(41)	91.83 (92.77)
Unbal. GPFAP	2	2.69 (2.74)	16.94 (17.13)	6.22 (6.23)	85.72 (85.90)
		2.60 (2.69)	16.2 (16.37)	**5.98 (6.13)**	**84.58 (85.13)**
	3	2.73 (2.75)	19.06 (19.17)	6.68 (6.77)	91.47 (91.94)
		2.63 (2.67)	16.83 (16.97)	6.42 (6.56)	89.51 (90.35)
	4	3.73 (3.85)	16.56 (16.72)	6.84 (6.88)	90.71 (91.52)
		2.85 (2.86)	**16.07 (16.27)**	6.75 (6.80)	90.02 (90.68)
	5	4.31 (4.38)	18.44 (18.57)	7.23 (7.24)	93.72 (94.04)
		3.28 (3.43)	16.30 (16.37)	6.83 (6.94)	91.18 (92.13)
Global		2.68 (2.76)	15.59 (15.64)	6.56 (6.62)	86.59 (87.12)

position. This result could be, however, predicted by considering that longer runs give the heuristic more opportunity to explore the search space, thus improving the actual performance of the whole approach. As a consequence, in some of the experiments with the highest number of subsets the decomposed approach appears to be effective only for the shortest runs (see Siemens1 with three and four subsets).

On the contrary, the GA seems still to show some significant improvements during the longest runs, see Figure 7.8. However, a further increment in the total number of evaluations will have a considerable effect on computational efficiency, since the GA tends to be slower than SA for the same number of configurations explored.

The decomposition usually produces better outcomes for a number of subsets usually between two and four, whereas further increments in their number degrade considerably the heuristic performance. This can be seen by the poor performance produced by a decompositions into five subsets. Although in some of the instances the unbalanced GPFAP produces better results than the balanced version, neither of

Table 7.9: GA - Siemens1-4 GA Balanced GPFAP decomposition ($2,000,000 * |V|$ evaluations per loop)

† at least 1 invalid solution * no valid solutions

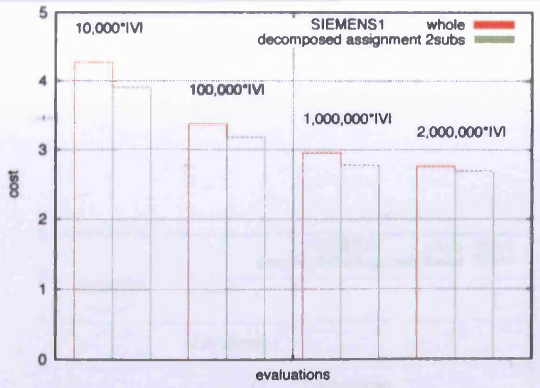
	noSub.	SIEMENS1	SIEMENS2	SIEMENS3	SIEMENS4
		first loop			
		second loop			
Bal. GPFAP	2	3.73 (3.80)	18.88 (19.03)	6.11 (7.65)	98.31 (99.93)
		3.40 (3.45)	**17.83 (17.86)**	**6.08 (7.21)**	**96.84 (97.22)**
	3	3.7 (3.75)	20.70 (20.95)	6.33 (7.09)	103.61 (104.40)
		3.30 (3.44)	18.70 (18.89)	6.21 (7.05)	101.09 (101.93)
	4	4.63 (4.81)	21.50 (21.79)	6.74 (7.34)	102.55 (103.60)
		3.37(3.49)	19.10 (19.21)	6.28 (7.19)	101.84 (102.19)
	5	5.70 (6.28)	21.07 (21.23)	7.32 (8.06)	110.71 (768.1) †
		4.45 (4.61)	18.95 (18.97)	6.66 (7.14)	106.81 (710.8) †
$2,000,000 * \|V\|$	2	3.53 (3.57)	18.96 (19.38)	6.34 (6.62)	104.49 (104.88)
		3.18 (3.28)	**17.46 (18.21)**	**6.17 (6.41)**	103.64 (103.68)
	3	3.31 (4.52)	20.57 (20.85)	7.02 (7.49)	103.11 (103.53)
		3.07 (3.14)	18.07 (18.44)	6.69 (6.77)	**102.89 (103.02)**
	4	3.46 (3.49)	18.87 (19.11)	7.37 (7.83)	103.52 (104.15)
		2.86 (3.04)	17.85 (18.23)	6.76 (6.88)	102.90 (103.24)
	5	3.75 (3.83)	20.03 (21.55)	7.43 (7.52)	104.71 (105.28)
		3.24 (3.27)	18.42 (18.79)	7.10 (7.06)	104.53 (104.84)
Global		3.61 (3.72)	17.64 (17.91)	9.54 (9.77)	101.05 (102.14)

the two algorithms appears able to clearly outperform the other. This is visualized in Figure 7.9 showing the histograms that compare, for different number of total evaluations, the best cost obtained by SA with decomposition with the balanced and unbalanced GPFAP with the one of the whole approach. Note that for the decomposed assignment approach the cost shown in the diagrams represents that obtained with the decomposition into the best performing number of subsets for each particular number of total evaluations considered. Similar behaviour is shown by the experiments conducted by the GA (see Appendix A).

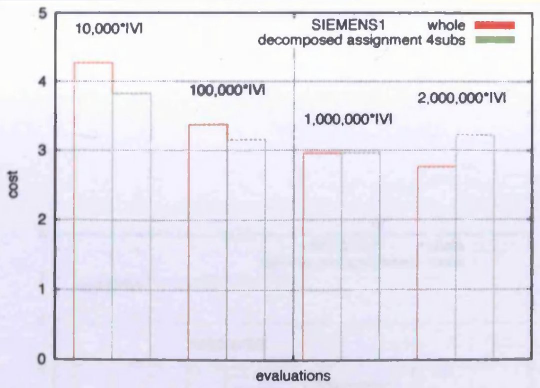
In general a simple decomposition into two subsets appears to be the most effective, although in some of the runs, and in particular for the unbalanced version, further decompositions can still improve the performance. Furthermore, our results improve for the standard SA those previously published for the same algorithm (see [2]), whereas no competitive results are known with GAs for the same benchmarks) However, this is not sufficient for improving the results of the solution produced by the whole approach in the case of Siemens2, which is the only benchmark for which the decomposition approach is ineffective on a pure quality basis. Another important aspect is that, although the GA produces overall a poorer performance than SA, its results become considerably better when the decomposed approach is applied. This actually reduces considerably the gap between the two algorithms (for example Siemens3). Histograms in Figures 7.9 and 7.10 compare the best cost produced by SA using the whole and by SA and GA using the decomposed approach. As above, for the decomposed approach the cost in the diagrams is that obtained with the best performing version of the GPFAP and number of subsets.

Finally, for both algorithms we can note the positive effect of the second loop, which becomes particularly important when the decomposition is initially not effective, for instance in the case of five subsets. This tendency will be confirmed by the other test problems considered for MI-FAP, in particular the Bradford benchmarks shown in the next subsection.

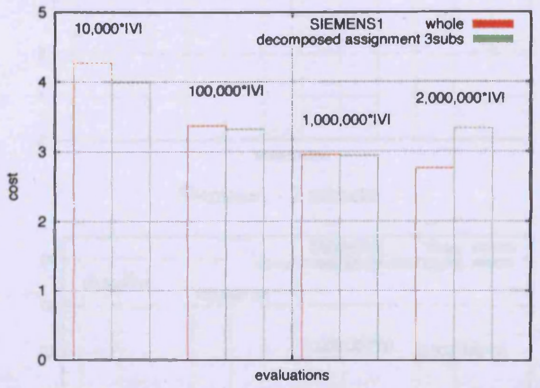
Figures 7.11 and 7.12 compares the cost-time curves produced for the Siemens1 and Siemens2 data sets by single runs of the meta-heuristics for a single loop with and without the decomposition technique. The solution of the decomposed assignment approach can be considered valid, that is consisting of a complete frequency assignment, only in the last of the subsets considered in the sequence (the second and the third represented with a dotted line in these examples). In these intervals, if we consider a fixed value of time the cost obtained by the decomposed assignment approach is generally lower than that produced by the whole approach. Note that this also happens in the cases in which the decomposed assignment approach appears ineffective (Siemens2), and also shows that a pure quality criterion may not be the most appropriate for evaluating the effectiveness of the decomposed approach.



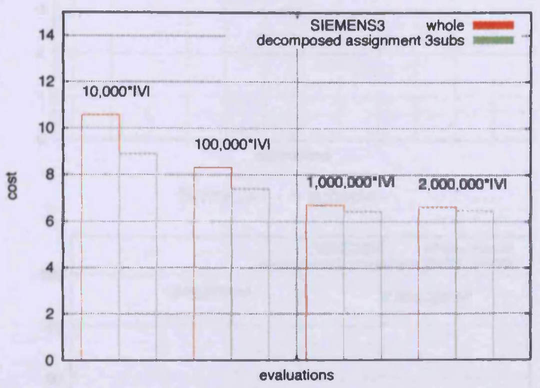
Siemens1 - 2 subsets



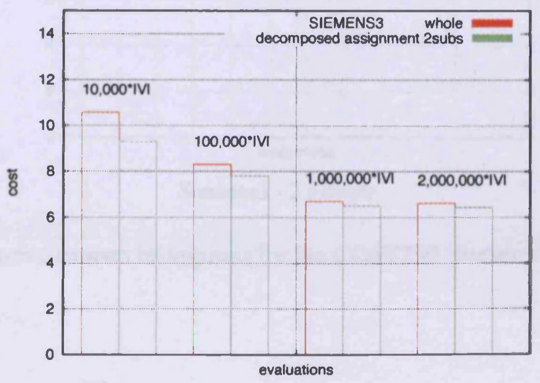
Siemens1 - 4 subsets



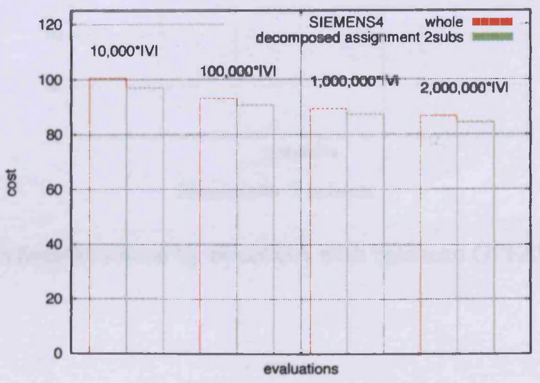
Siemens1 - 3 subsets



Siemens3 - 3 subsets



Siemens3 - 2 subsets



Siemens4 - 2 subsets

Figure 7.7: Cost-evaluations histograms for the COST259 Siemens benchmarks solved by SA with balanced GPFAP decomposition

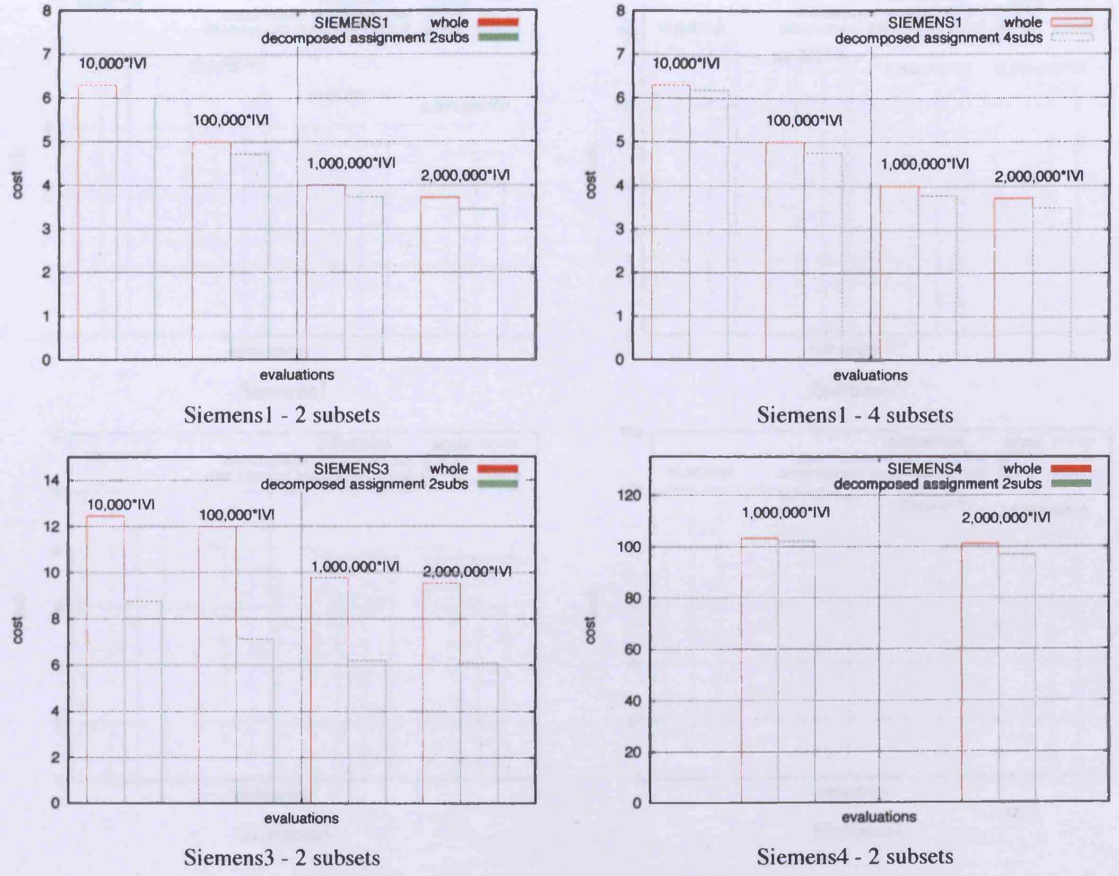
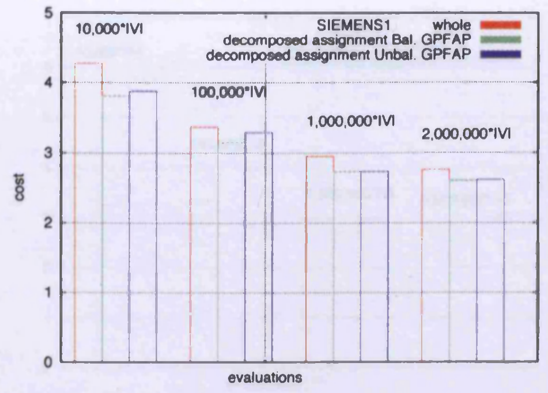
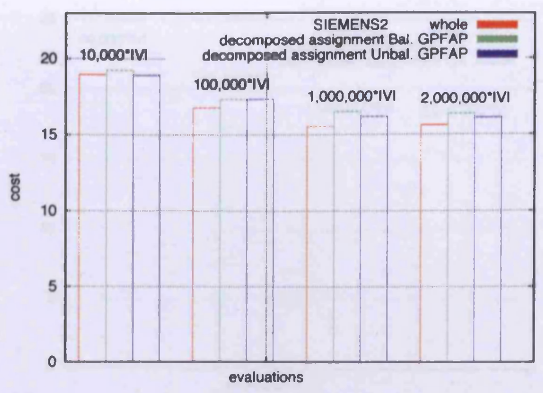


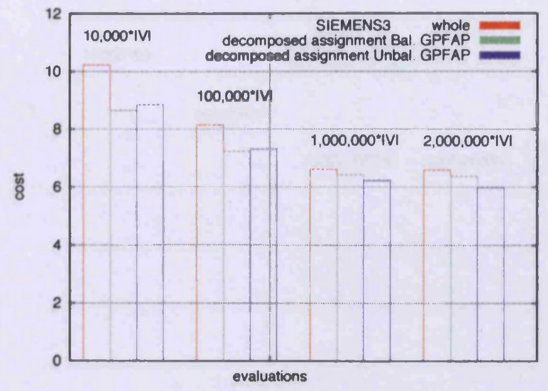
Figure 7.8: Cost-evaluations histograms for the COST259 Siemens benchmarks solved by direct GA with balanced GPFAP decomposition



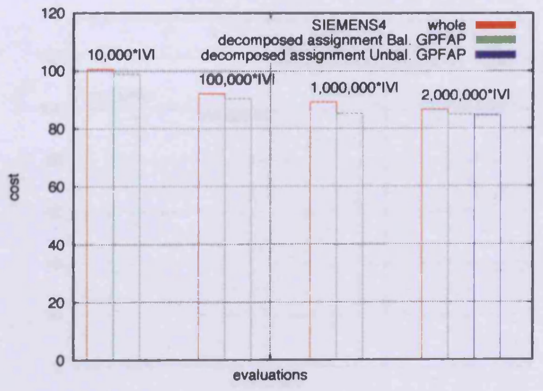
Siemens1



Siemens2



Siemens3



Siemens4

Figure 7.9: Cost-evaluations histograms for the COST259 Siemens benchmarks solved by the whole approach (SA) and balanced\unbalanced GPFAP (SA and GA)

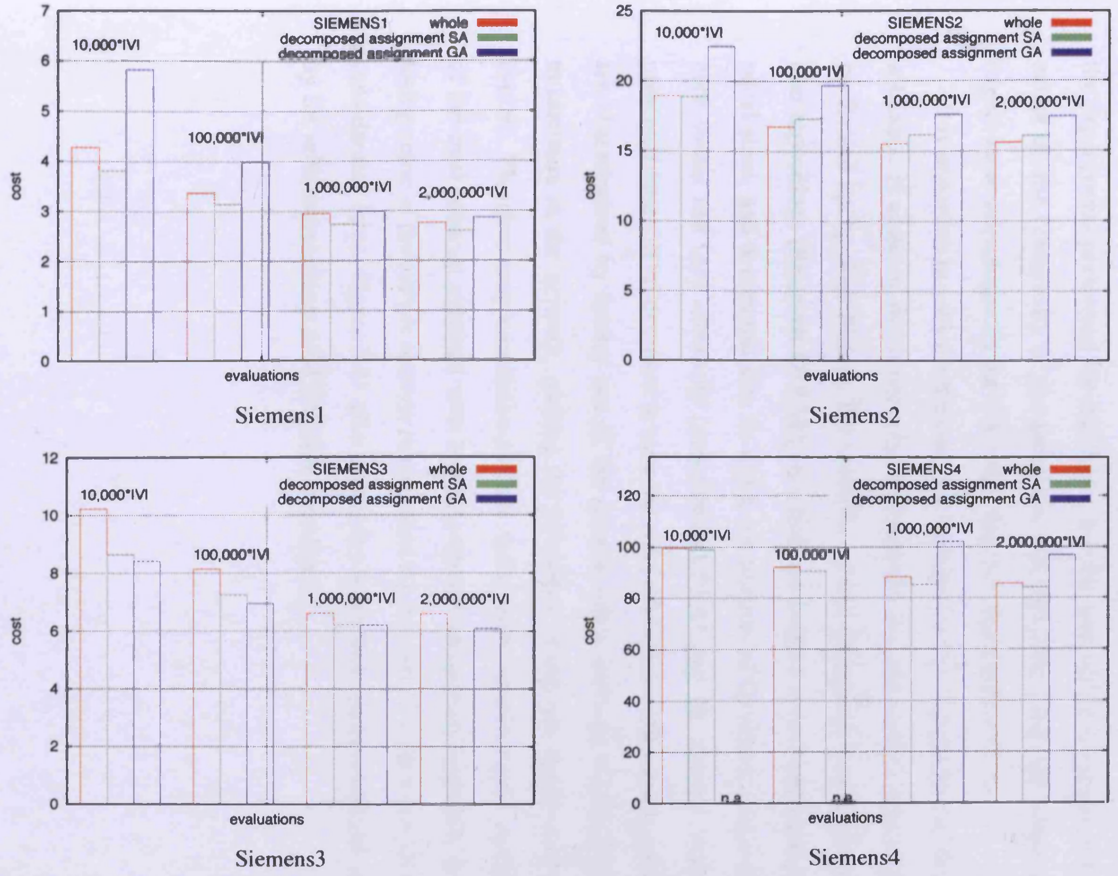
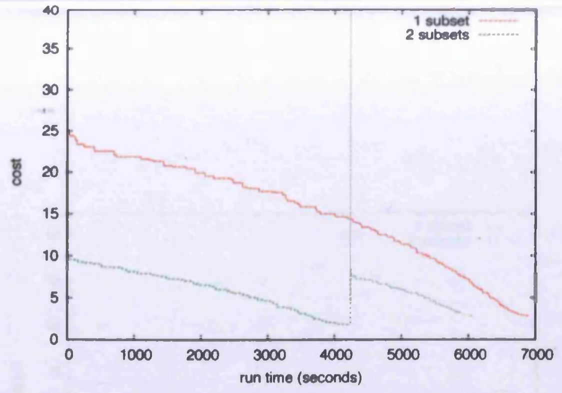


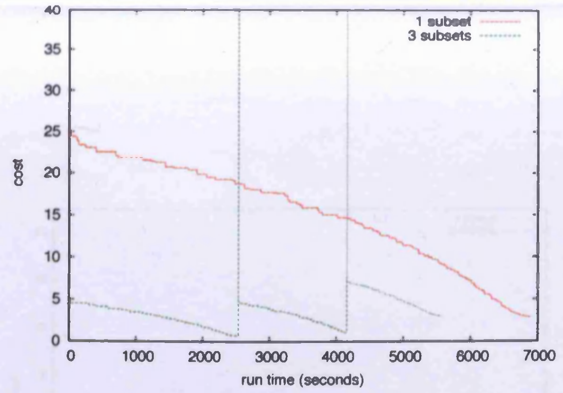
Figure 7.10: Cost-evaluations histograms for the COST259 Siemens benchmarks solved by the whole approach (SA) and balanced\unbalanced GPFAP (SA and GA)

Both curves obtained with the whole and the decomposed approach correspond to the same final number of total evaluations conducted by the meta-heuristic. With this constraint we can observe that the decomposition approach, independently of being more or less optimal, is able to produce very good approximations in a shorter time. The runtime gain can be empirically quantified in a 10 – 15% gain on the experiments performed for this thesis, but this amount does depend to a large extent on the complexity of the graph and the algorithm used (for instance, we expect more advantages by the GA rather than SA, see Section 4.1.2).

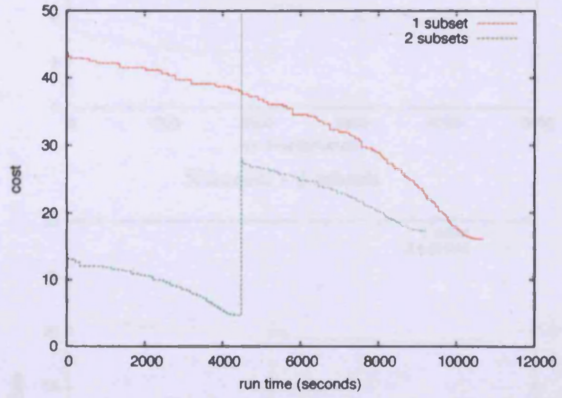
It is important to consider the trade-off between quality and runtime of the final solutions. In addition, this is not only influenced by the total number of evaluations performed but by other factors. For example, we can distinguish between balanced decompositions (balanced GPFAP), in which the subsets are of approximatively equal sizes, and decompositions in which the number of transmitters included in each subset can vary arbitrarily (unbalanced GPFAP and the Markov clustering presented later). It is important to note that very often these unbalanced partitions are characterized by having one of the subsets which includes nearly all of the transmitters in the network, making the behaviour of the two approaches more similar. This fact may sometimes produce some improvements in the optimality of the final solution obtained with the decomposed assignment approach despite losing some of the gain in runtime terms, since the interval of valid solutions starts considerably later. Figure 7.11 also shows the same Siemens benchmarks solved by SA with an unbalanced GPFAP into two subsets.



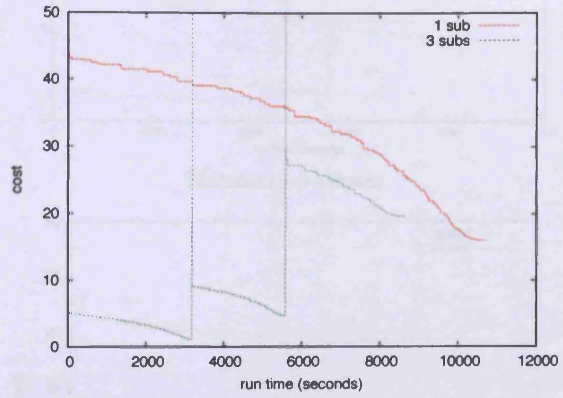
Siemens1 - balanced - 2 subsets



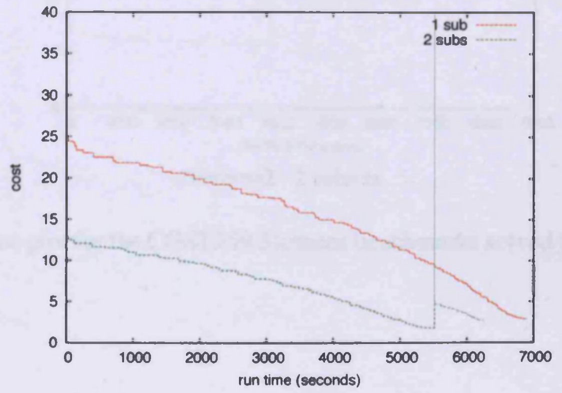
Siemens1 - balanced - 3 subsets



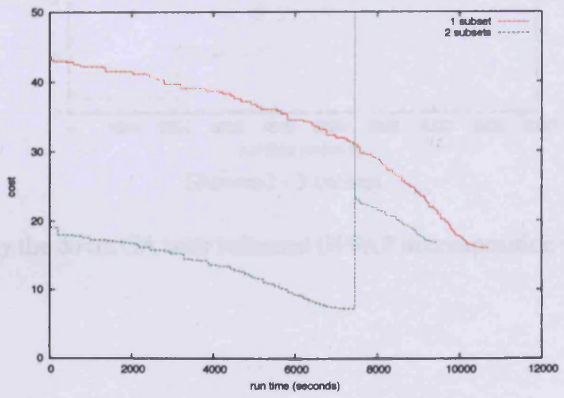
Siemens2 - balanced - 2 subsets



Siemens2 - balanced - 3 subsets



Siemens1 - unbalanced - 2 subsets



Siemens2 - unbalanced - 2 subsets

Figure 7.11: Cost-time plot for the COST259 Siemens benchmarks solved by SA with GPFAP decomposition after $2,000,000 * |V|$ evaluations

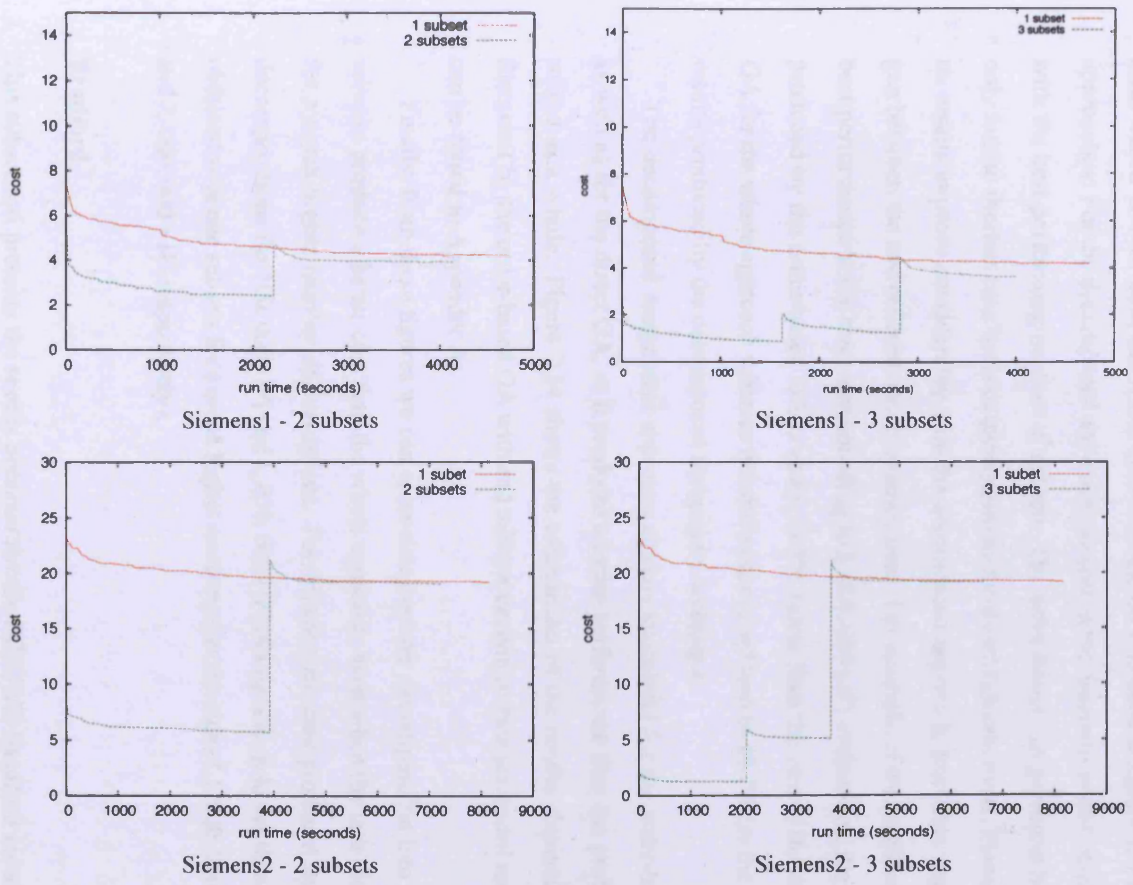


Figure 7.12: Cost-time plot for the COST259 Siemens benchmarks solved by the direct GA with balanced GPFAP decomposition after $2,000,000 * |V|$ evaluations

As anticipated in Section 6.3, because of the inherent limitations of its representation, the order-based GA produces a poor performance when it is applied in the traditional way of solving the problem as a whole. Figure 7.13 shows a comparison between the average cost over three runs produced by the direct and the order-based GA for both the whole (left) and the decomposed assignment (right) approaches. For the decomposed approach the cost in the diagrams is that obtained with the best performing number of subsets. The order-based GA performs better only for the shortest runs but is outperformed by the direct GA otherwise. However, its results improve considerably with the decomposed approach, thus reducing the gap between the two different representation used. For example, if we consider the best performance in the runs corresponding to $2,000,000 * |V|$ evaluations, the cost produced by the order-based GA is about 29.8% worse than the one of the direct GA for the whole approach whereas this difference is reduced to 10.7% for the best results produced by the decomposed assignment technique.

The decomposed assignment approach appears successful for the order-based as well as for the direct GA, as it produces a better performance than the problem solved as a whole. Figure 7.14 shows the comparison of the results obtained for Siemens1 by the order-based GA with and without decomposition (extended results can be found in Appendix A).

Finally, from these figures we can appreciate how the decomposition into subsets can produce a better cost than the whole approach, even when the latter is run for a much higher number of evaluations. For instance the cost produced by the decompositions for $100,000 * |V|$ and $1,000,000 * |V|$ evaluations is lower than that obtained with one subsets for a much higher number of evaluations ($1,000,000 * |V|$ and $2,000,000 * |V|$ respectively).

Bradford

This subsection presents the results obtained for the COST259 Bradford instances solved by the balanced GPFAP for short and long runs, that is $100,000 * |V|$ and within the network caused by the second loop, whose effect actually balances the constraint violations, and so the local interference, between different subsets. In addition we note that the decomposition technique is mostly effective with a minimum number of two subsets (with again the exception of Bradford0), thus confirming that these instances appear harder than the Siemens benchmarks in order to be solved with the decomposed approach.

Figures 7.15 shows the cost-time plot produced by the decomposed assignment approach for the Bradford0 benchmark for balanced and unbalanced GPFAP. Note that for this benchmark this approach is successful for both the first and second loop through the subsets. Finally, Figure 7.16 shows the behaviour of Bradford0, Bradford4, and Bradford10, for which the decomposition is only effective at the end of the second loop, that is at the end of the first loop it still produces worse results than the whole approach for the same number of evaluations.

Partitioning on soft constraints only

For a representative pair of COST259 benchmarks (Siemens1 and Bradford0), we conducted further experiments in which the partitioning was obtained using different values of the weights λ_i used in the Definition 7.2 in order to define the weighted simple graph G^D. In particular we set to null values the weights concerning the hard constraints, which will therefore be ignored in the process defining the partitioning of the interference graph.

Note that, for these instances, there are always corresponding soft constraints for co-channel and adj-channel interference wherever there is a hard co-site or handover constraint. However, this is not true for the hard co-cell constraints. As a consequence, although useful in showing the eventual dependance between the decomposed assignment approach and the specific type of constraints, these experiments can be considered slightly unrealistic. However, this will not affect the results for the GPFAP since applying the cellular form of decomposition automatically preserves the hard co-cell constraints as observed in 7.2.1.

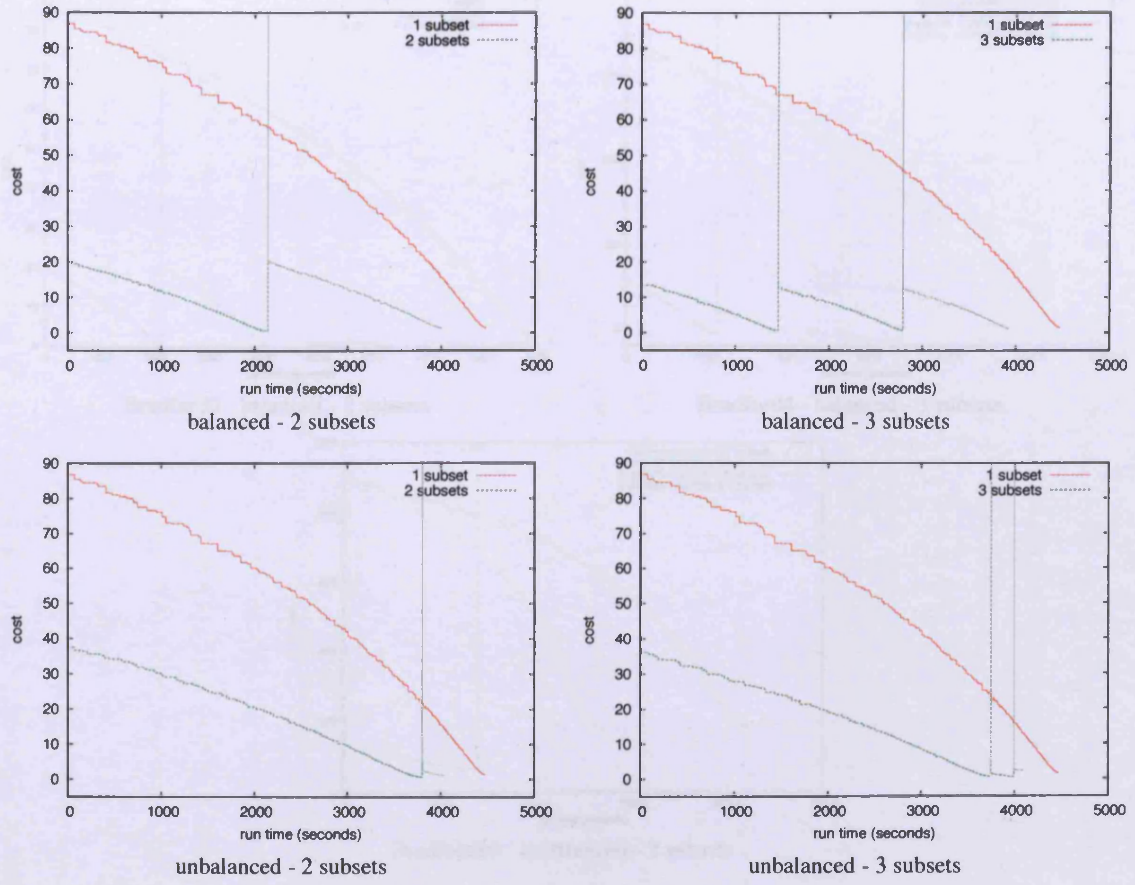
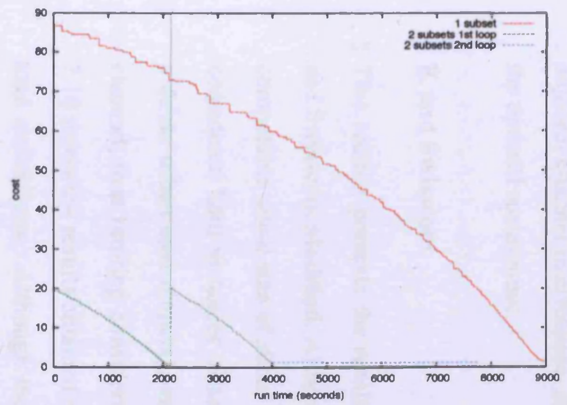
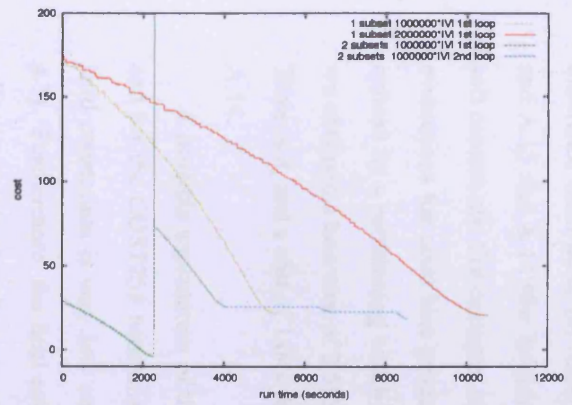


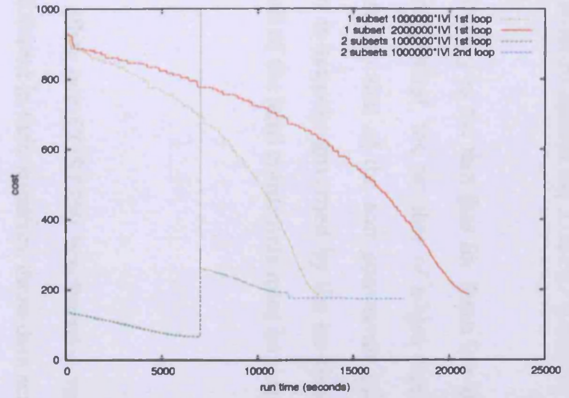
Figure 7.15: Cost-time plot for Bradford0 solved by SA with GPFAP decomposition after $1,000,000 * |V|$ evaluations



Bradford0 - balanced - 2 subsets



Bradford4 - balanced - 3 subsets



Bradford10 - unbalanced - 2 subsets

Figure 7.16: Cost-time plot for the COST259 Bradford instances solved by SA with balanced GPFAP decomposition with two loops ($1,000,000 * |V$ evaluations per loop)

We performed experiments using SA with balanced GPFAP for a variable number of evaluations between $10,000 * |V|$ and $2,000,000 * |V|$ total evaluations (see Appendix A for a summary of the results). The decomposition still improves the results produced by the whole approach, although they do not show any significant difference from those previously reported in Tables A.7 and A.8 (for Siemens1), and A.15 and A.14 (for Bradford0) with a different combination of the hard and soft constraints. For example, if we consider the longest runs with $2,000,000 * |V|$ evaluations the costs here produced are only a slight improvement over those calculated by a partitioning which considers both hard and soft constraints. In detail, we obtained a best cost of 2.598 for Siemens1 against one of 2.601 (– 0.001%) in Table A.8, and a cost of 1.044 for Bradford0 against the 1.067 (– 0.022%) in Table A.14.

A possible explanation of this may lay in the fact that for these two data sets, and for the COST259 benchmarks in general, the number of edges representing hard constraints is very low compared to that of the soft constraints (see Table 4.7). Furthermore, the final solution is actually governed by the co-channel and adjacent channel interference, since all of the hard constraints must be respected in the optimal assignment.

K and Swisscom

This section presents the results of other two COST259 benchmarks, namely *K* and Swisscom Modified. Although different in their structure these data sets have a comparable small size of about three hundred vertices. However, they can be both considered hard instances since *K* presents a very high connectivity (simulating a dense urban environment), and Swisscom Modified a high number of blocked channel, thus limiting considerably the spectrum of frequencies available. Table 7.10 shows the results obtained with GPFAP for $3,000,000*|V|$ and $5,000,000*|V|$ total evaluations. Although the results presented in [2] for Swisscom Modified refer to a partial assignment that need to be completed (so some of the transmitters are assigned a fixed frequency), we have removed in our experiments this further constraint, in order to keep consistency with the other experiments presented in this chapter.

Table 7.10: SA - K and Swisscom Modified with GPFAP decomposition

† at least 1 invalid solution * no valid solutions

noEvals.	noSub.	K		SWISSCOM MODIFIED2	
		Bal. GPFAP	Unbal. GPFAP	Bal. GPFAP	Unbal. GPFAP
		first loop			
		second loop			
3,000,000 • $\|V\|$	1	0.98 (1.09)		34.06 (34.43)	
		0.84 (0.89)		32.24 (32.34)	
	2	1.18 (1.24)	0.82 (0.85)	30.58 (2,031)	29.37 (30.57)
		0.77 (0.86)	0.72 (0.75)	29.31 (30.51)	**28.78 (28.99)**
	3	1.41 (1.44)	0.80 (0.84)	2,029 (4,030)	30.62 (30.85)
		1.03 (1.17)	**0.71 (0.72)**	2,031 (3,031)	29.01 (29.25)
	4	1.80 (1.88)	0.85 (0.87)	2,032 (5,033)	33.30 (33.84)
		1.15 (1.19)	0.75 (0.78)	2,029 (5,025)	32.73 (33.14)
5,000,000 • $\|V\|$	1	0.85 (0.87)		33.31 (33.49)	
		0.81 (0.84)		30.73 (31.25)	
	2	1.12 (1.19)	0.64 (0.66)	29.39 (30.30)	29.27 (30.14)
		0.63 (0.65)	**0.61 (0.57)**	27.09 (27.80)	**26.04 (26.78)**
	3	1.27 (1.40)	0.66 (0.70)	30.98 (2,032)	29.54 (29.86)
		0.81 (0.88)	0.64 (0.69)	29.12 (1,026)	27.10 (27.51)
	4	1.65 (1.70)	0.72 (0.74)	6,030 (4,030)	32.46 (32.57)
		0.97 (1.01)	0.69 (0.71)	5,031 (3,035)	32.12 (32.20)

For K the decomposed assignment approach with the unbalanced version of the GPFAP is effective both at the end of the first and second loop (for a maximum improvement compared to the whole approach of 35% and 33% respectively for the longest run). On the contrary, the balanced GPFAP improves on the whole approach only after the second loop. This is caused by the very high density of the graph (see Table 4.7), which makes necessary a redistribution of the interference during the second loop in order to produce effective results. Note that, at least for decomposition into two subsets, the balanced and unbalanced GPFAP eventually produce comparable results after the two loops (for a maximum improvement for the balanced version of 28% for the longest run).

For Swisscom Modified, because of the limited spectrum availability, the shortest runs have difficulty in producing feasible assignment during the runs with the decomposed assignment approach, thus it became necessary to increase the number of total evaluations. However, the balanced decomposition still produces infeasible solutions when decomposed in more than two subsets, whereas the unbalanced GPFAP appears always able to produce assignments belonging to the feasibility domain. When the solutions produced are feasible, the decomposed assignment outperforms the whole approach (for a best improvement of 18% produced by the unbalanced version for the longest run).

Cardiff University benchmarks

This section presents the results obtained by the GPFAP on the large benchmarks generated by Cardiff University. Note that for very large benchmarks the decomposition approach is expected to be effective, since this represents a case in which the performance of the meta-heuristics starts degrading, at least in their standard versions. Table 7.11 reports the costs produced by SA with the balanced GPFAP for $100,000 * |V|$ and $2,000,000 * |V|$ total evaluatons. We have also considered the geographical decomposition since it represents an intuitive and natural form of partitioning for these benchmarks in which transmitters are located in 'towns'.

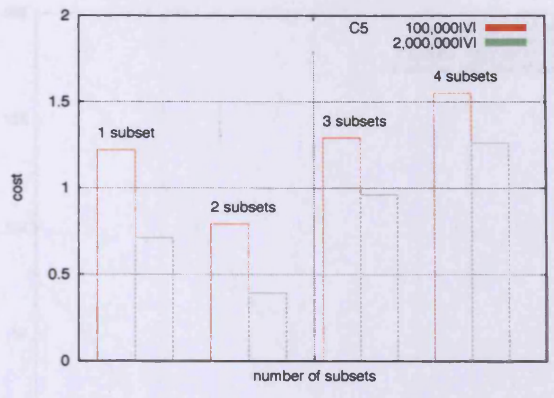
Table 7.11: SA - Cardiff University benchmarks with balanced GPFAP and geographical decomposition

noEvals.	noSub.	C_5		C_6	
		Geog	GPFAP	Geog	GPFAP
		first loop			
		second loop			
$100,000 * \|V\|$	1	2.16 (2.31)		1.18 (1.30)	
		1.22 (1.32)		0.73 (0.89)	
	2	1.68 (1.82)	1.30 (1.75)	0.62 (0.75)	0.51 (0.63)
		0.98 (1.14)	**0.79 (0.93)**	0.28 (0.42)	**0.23 (0.31)**
	3	3.03 (3.46)	2.72 (3.01)	1.25 (1.52)	0.94 (0.99)
		1.43 (1.76)	1.29 (1.38)	0.43 (0.65)	0.32 (0.36)
	4	3.37 (3.57)	3.18 (3.25)	1.39 (1.85)	1.09 (1.13)
		1.76 (2.11)	1.55 (1.68)	0.55 (0.67)	0.42 (0.56)
$2,000,000 * \|V\|$	1	0.95 (1.04)		0.37 (0.42)	
		0.71 (0.74)		0.31 (0.36)	
	2	1.01 (1.25)	0.91 (1.16)	0.30 (0.34)	0.23 (0.29)
		0.59 (0.68)	**0.39 (0.42)**	0.09 (0.12)	**0.06 (0.10)**
	3	2.47 (2.71)	2.14 (2.25)	0.51 (0.57)	0.50 (0.53)
		1.10 (1.60)	0.96 (1.09)	0.16 (0.21)	0.14 (0.19)
	4	2.65 (2.84)	2.27 (2.34)	0.57 (0.77)	0.56 (0.63)
		1.52 (1.66)	1.26 (1.41)	0.28 (0.46)	0.25 (0.35)

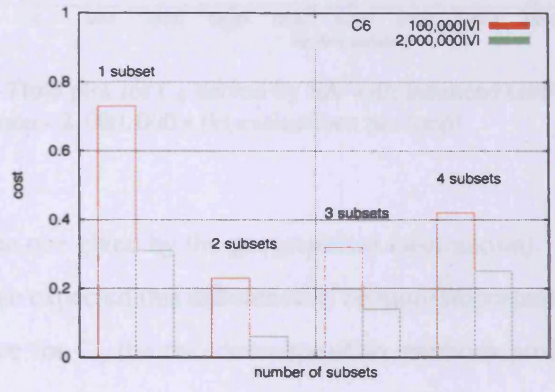
Figure 7.17 shows the histogram of best cost (after two loops) against number of total evaluations for different number of subsets. There is a marked difference between the values produced by the shortest and longest runs, thus confirming that large benchmarks need to be run for a longer time. Furthermore, this tendency is more important for the whole approach whereas the decomposed assignment approach is able (when effective) to produce good results also for the shortest runs. For the decomposed approach the results still improve if we increase the number of evaluations, although the percentage improvement in comparison to the whole approach is lower than for the shortest runs. This is particularly true for C_6 with a decomposition into two subsets, for which the shortest run with $100,000 * |V|$ evaluations outperforms that obtained with the whole approach for $2,000,000 * |V|$ evaluations. However, in any case the shortest runs give an indication of whether or not the decomposed approach will be effective.

Histograms in Figure 7.19 compare, for the different number of evaluations, the best costs produced at the end of each loop for different number of subsets. A decomposition into two subsets always improves the results of the whole approach, although it produces better performance during the second loop. Figure 7.18 shows the cost-time behaviour for a single run with two loops of the 'two-towns' benchmark C_6 ($2,000,000 * |V|$ evaluations per loop). At the end of each loop the diagram compares the solution produced by the decomposed assignment approach with that of the whole approach for the same number of total evaluations.

For a decomposition into a larger number of subsets (three and four in our examples), the decomposed technique is generally not effective after the first loop through the subsets. However, the second loop brings about a remarkable improvement leading in some cases to better results than the whole approach (see C_6). Note that this happens either for the shortest or the longest runs, thus confirming the validity of the former ones as a test for the effectiveness of a given decomposition. For the whole approach the gap between the first and the second loop is more marked in the shorter runs whereas for the longer runs it is less significant. The



C_5



C_6

Figure 7.17: Cost-number of evaluations histograms for C_5 and C_6 solved by SA with balanced GPFAP

decomposed approach still shows a marked improvement between the two runs independently from their duration. Moreover, this improvement is more significant when the effectiveness of the decomposition is worse (that is for three and four subsets).

Figures 7.20 shows a comparison between GPFAP and geographical decomposition for different number of subsets and total evaluations. The decomposition based on geographical information is inferior for both of the benchmarks. This can be also interpreted as the confirmation of the effectiveness of the GPFAP decomposition procedure, which is able to find an effective partition without any extra

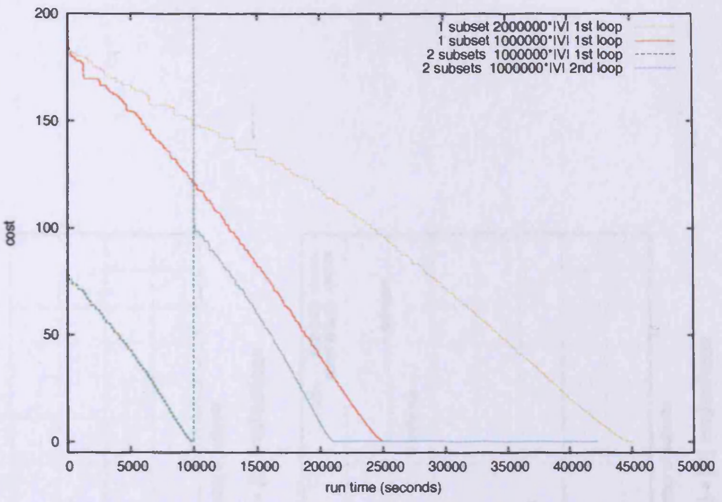
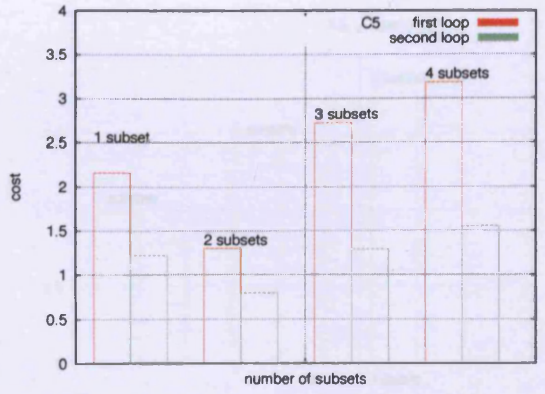


Figure 7.18: Cost-Time plot for C_6 solved by SA with balanced GPFAP into two subsets (first and second loop - $2,000,000 * |V|$ evaluations per loop)

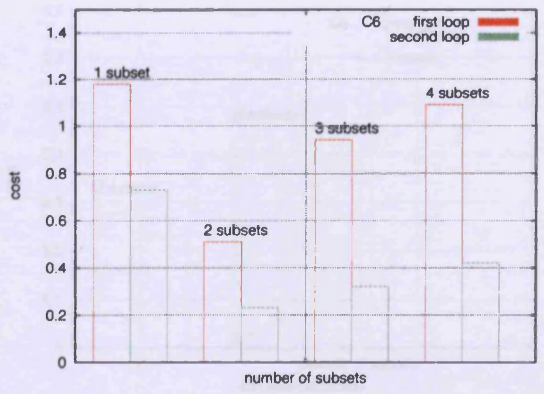
knowledge (as the one given by the geographical information).

We could have expected this difference to be more important for the 'one town' problem C_5, since for C_6 the two decomposition methods produce rather similar partitions into the two distinct towns (see Figure 7.5). However, this actually happens only for the longest runs, whereas the two decompositions show similar behaviour with both of the data sets for the runs with $100,000 * |V|$ evaluations.

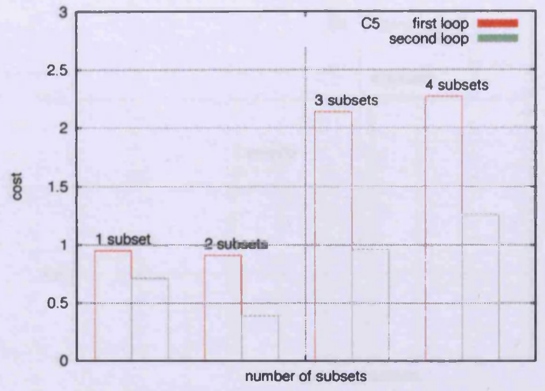
Figure 7.21 shows the cost- number of evaluations plots for two single runs of the benchmarks C_5 and C_6, in which it can be appreciated how the shortest runs produce results which are very similar to those obtained for a much higher number of evaluations. Moreover, we can note that for the 'two-towns' problem C_6 the first subset still produces significant interference whereas for the 'one-town' C_5 the assignments in the first subset are almost interference free during the whole length of its run.



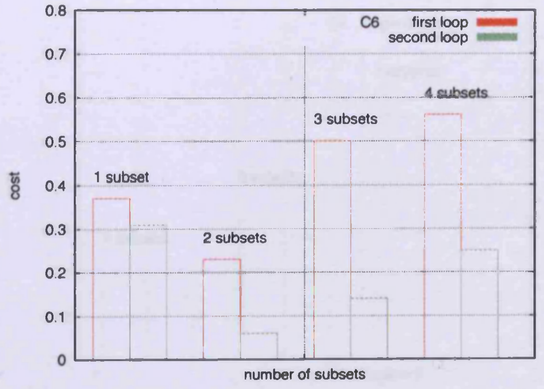
C_5 - $100,000 * |V|$ evaluations



C_6 - $100,000 * |V|$ evaluations

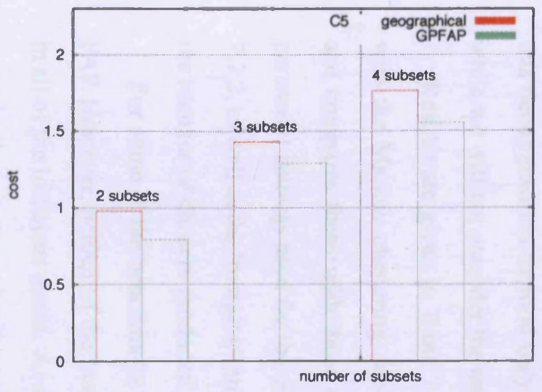


C_5 - $2,000,000 * |V|$ evaluations

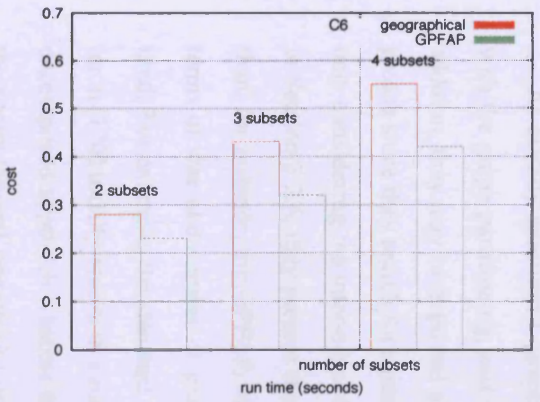


C_6 - $2,000,000 * |V|$ evaluations

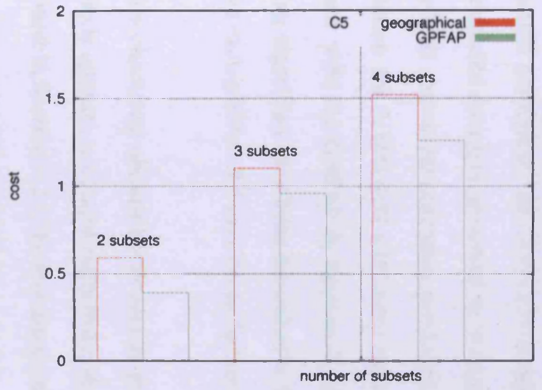
Figure 7.19: Cost-number of subsets histograms for C_5 and C_6 solved by SA with balanced GPFAP with two loops



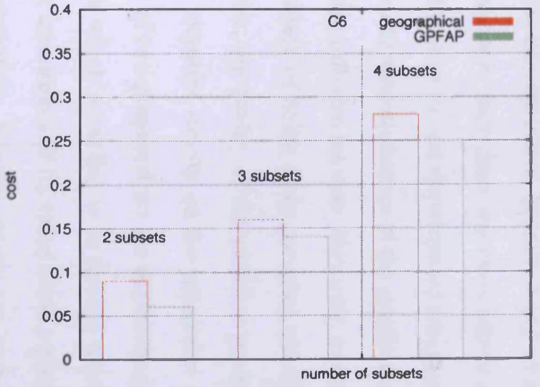
C_5 - 100, 000 * |V| evaluations



C_6 - 100, 000 * |V| evaluations



C_5 - 2, 000, 000 * |V| evaluations



C_6 - 2, 000, 000 * |V| evaluations

Figure 7.20: Comparison between GPFAP and geographical decomposition for C_5 and C_6

7.4 Markov clustering

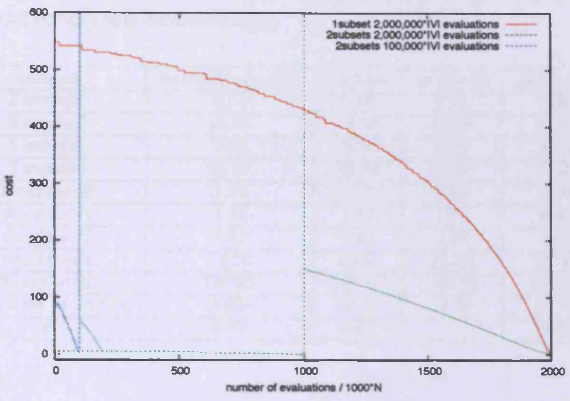
To complete the set of the experiments for the MI-FAP instances, we have conducted a number of runs which use the partitions generated by the Markov clustering procedure (see Section 3.2.6).

As already mentioned, graph clustering algorithms present many similarities with the graph partitioning, and consequently they share the main advantages. In addition, they may be expected to be effective for the decomposed assignment approach since they search for a partition into natural clusters of the graphs instead of only considering the inter-edges between different subsets. However, as described in Section 3.2.6, they present the drawback of being more complex and elaborate than, for instance, the GPFAP. Moreover, the quality of the partition produced, in terms of size and number of clusters, depends strictly on the parameter settings used. For example, the standard value of two proposed for the expansion parameter in [120] usually results in a partition which is too fine to be directly used for the decomposed approach whereas further increments of its value affect considerably the computational complexity of the algorithm. For this reason we have applied this decomposition method only to a small number of the COST259 benchmarks while we will not consider the larger generated data sets presented in Section 4.2.2.

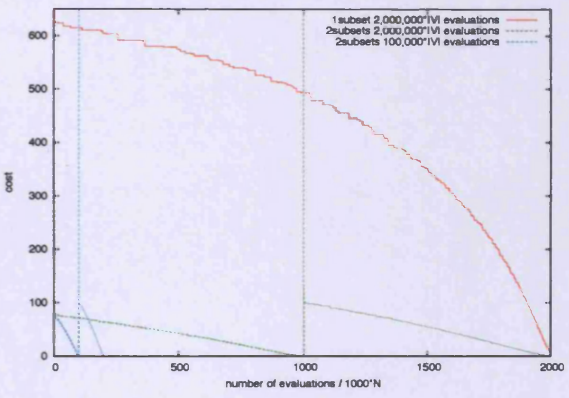
Results are given in Table 7.13 , which shows the outcomes produced by SA with the Markov clustering decomposition for $2,000,000 * |V|$ total evaluations and compares them with those obtained with the GPFAP in Section 7.3.2. The parameter settings used for the clustering algorithm have been described in Section 7.2.2, as well as the 'merging' procedure in Algorithm 7.5 in order to further reduce the number of clusters produced.

For some of the benchmarks Markov clustering obtains better results than GPFAP. However, neither of the two methods appears to clearly outperform the other in all of the instances tested. As mentioned in Section 7.2.2 the resulting clustering is normally unbalanced in size and so it is expected to produce outcomes which are close to those of the unbalanced GPFAP (see also Tables 7.8, A.14, and A.16).

This is confirmed by the results shown in Table 7.13. Note that Markov clustering is not able to obtain positive results for the data sets with the highest graph density (i.e. K and Siemens2), for which it performs worse than GPFAP. On the contrary, its performance improves for the other Siemens benchmarks which show a more 'natural' partition into clustering, as shown by the distribution of the hard constrains in Figure 7.6.



C_5



C_6

Figure 7.21: Cost-Number of evaluations plot for C_5 and C_6 solved by SA with balanced graph partitioning decomposition into two subsets

To summarise, before having a look to some issues about the distribution of the resulting interference in the network, Table 7.12 shows the best results obtained by the decomposition approach with 9SA over all the different decompositions tried, as well as those of the global approach and (for comparison). We also show the state-of-the-art best costs produced for the COST259 benchmarks by ether SA or other meta-heuristicsas reported in [2]. Results for our implementations are those obtained with $2,000,000*$ evaluations. For Swisscom the best results from [2] refers to the original version and not o the modified one used for this thesis (although the interference constraints for the two versions are identical).

Table 7.12: Comparison between the best costs produced by the SA with and without decomposition and other meta-heuristics [2]

benchmark	SA with decomposition	SA global	SA from [2]	Best from [2]
*Siemens*1	2.60	2.68	2.78	**2.20**
*Siemens*2	16.07	15.59	15.46	**14.27**
*Siemens*3	5.98	6.59	6.75	**4.73**
*Siemens*4	84.08	86.59	89.15	**77.25**
*Bradford*0	1.04	1.31	0.80	**0.60**
*Bradford*1	1.32	1.64	1.04	**0.86**
*Bradford*2	3.83	4.46	3.79	**3.20**
*Bradford*4	18.51	20.62	19.00	**17.72**
*Bradford*10	185.05	187.13	148.12	**144.94**
Swisscom Modified	26.04	30.73	27.36	**17.71**
K	0.61	0.81	–	**0.45**

Table 7.13: COST259 for SA with GPFAP and Markov clustering decomposition - best(average) cost using $2,000,000 * |V|$ evaluations per loop and $5,000,000 * |V|$ evaluations per loop.

ns	GPFAP$_{bal}$	GPFAP$_{unb}$	Markov	GPFAP$_{bal}$	GPFAP$_{unb}$	Markov
	first loop					
	second loop					
	SIEMENS1			SIEMENS2		
1	2.75(2.83)			15.72(15.79)		
	2.68(2.76)			**15.59(15.54)**		
2	2.68(2.75)	2.69(2.74)	2.76(2.92)	16.97(17.04)	16.94(17.13)	19.71(19.89)
	2.60(2.69)	2.61(2.66)	**2.53(2.64)**	16.34(16.73)	16.20(16.37)	17.32(17.58)
3	2.95(3.01)	2.67(2.75)	2.99(3.14)	19.54(19.77)	19.06(19.17)	20.22(20.45)
	2.93(2.95)	2.63(2.72)	2.74(2.83)	17.41(17.60)	16.83(16.97)	17.90(18.47)
4	2.94(3.01)	3.73(3.85)	3.08(3.16)	20.56(20.87)	16.56(16.72)	21.97(21.99)
	2.90(2.98)	2.85(2.87)	2.77(2.79)	18.02(18.27)	16.07(16.27)	19.59(19.90)
5	3.34(3.43)	4.31(4.48)	3.76(3.99)	20.31(20.46)	18.46(18.57)	22.36(22.97)
	3.05(3.13)	3.29(3.44)	2.77(2.94)	17.91(18.53)	16.30(16.37)	19.59(20.04)
	SIEMENS3			SIEMENS4		
1	6.61(6.72)			87.25(87.98)		
	6.59(6.62)			86.59(87.12)		
2	6.39(6.46)	6.22(6.24)	6.528(6.88)	84.35(84.80)	85.72(85.90)	83.67(83.91)
	6.37(6.44)	**5.98(6.13)**	6.24(6.35)	84.08(84.39)	84.58(85.13)	**82.37(82.54)**
3	6.53(6.81)	6.68(6.77)	6.44(6.63)	89.50(90.59)	91.47(91.94)	85.12(86.40)
	6.46(6.58)	6.42(6.56)	6.18(6.39)	87.16(88.51)	89.51(90.35)	84.70(85.27)
4	6.95(7.02)	6.84(6.88)	7.12(7.37)	89.96(90.43)	90.71(91.52)	93.20(94.05)
	6.79(6.89)	6.76(6.80)	7.01(7.49)	89.53(90.17)	90.02(90.68)	91.32(92.88)
5	7.65(7.79)	7.23(7.25)	8.71(9.15)	92.94(93.55)	93.72(94.04)	94.08(95.12)
	7.26(7.41)	6.83(6.94)	7.51(8.24)	91.83(92.77)	91.18(92.13)	92.61(93.45)
	BRADFORD0			BRADFORD4		
1	1.42(1.66)			20.84(21.07)		
	1.31(1.43			20.62(20.79)		
2	1.31(1.41)	1.37(1.72)	1.04(1.10)	21.40(21.83)	22.17(22.77)	22.90(23.16)
	1.16(1.23)	1.08(1.14)	0.94(1.11)	**18.51(18.78)**	19.95(20.44)	19.55(20.22)
3	1.18(1.20)	1.11(1.19)	1.03(1.08)	25.83(26.27)	26.64(27.81)	27.29(27.54)
	1.07(1.13)	1.09(1.13)	**0.80(1.06)**	20.54(21.01)	21.66(22.21)	22.71(23.16)
4	1.48(1.63)	2.17(2.31)	2.03(2.17)	33.93(34.29)	34.82(35.29)	34.98(35.43)
	1.26(1.36)	1.24(1.26)	1.39(1.52)	23.09(24.65)	24.63(25.50)	25.23(25.58)
5	2.07(2.15)	2.61(2.72)	2.17(2.55)	37.85(38.14)	38.71(39.31)	37.31(37.98)
	1.47(1.54)	1.26(1.29)	1.31(1.52)	28.52(29.83)	29.68(30.56)	28.83(29.59)
	BRADFORD2			K *		
1	4.56(4.71)			0.85(0.87)		
	4.46(5.53)			0.81(0.84)		
2	5.17 (5.36)	4.79 (4.99)	5.32 (5.51)	1.12 (1.19)	0.64 (0.66)	1.47 (1.98)
	4.27 (4.24)	**3.83 (3.92)**	4.09 (4.23)	0.63 (0.65)	**0.61 (0.57)**	0.95 (1.02)
3	6.45 (6.92)	6.15 (6.21)	5.98 (6.30)	1.27 (1.40)	0.66 (0.70)	2.51 (3.02)
	5.34 (5.49)	5.05 (5.24)	4.22 (4.85)	0.81 (0.88)	0.64 (0.69)	1.08 (1.15)
4	6.97 (7.17)	6.25 (6.34)	5.69 (6.01)	1.65 (1.70)	0.72 (0.74)	2.76 (3.27)
	5.75 (5.93)	5.14 (5.33)	4.46 (4.89)	0.97 (1.01)	0.69 (0.71)	1.12 (1.53)
5	7.25 (7.64)	6.79 (7.24)	5.94 (6.15)	2.04 (2.32)	1.05 (1.33)	2.74 (2.95)
	6.13 (6.58)	5.96 (6.57)	5.03 (5.65)	1.79 (1.96)	0.94 (1.12)	1.12 (1.50)

7.5 Trade-off between quality and runtime

As described in Section 4.1.3, given a specific benchmark and decomposition method we evaluate the trade-off between quality and runtime by plotting the interpolation curves between the pairs {cost (best or mean), runtime} for a number of different runs of the decomposed assignment approach.

We fix a number of total evaluations and then we run the decomposed approach with a different number of subsets, together with the whole one for the same number of evaluations. By considering only one loop and according to the parameters setting in Section 7.1 we then explore $eval_{i,0} = \frac{eval_{tot}*|V_i|}{|V|}$ per subset, so that the number of evaluations, in each of the subsets is reduced proportionally to its cardinality.

This is expected to produce roughly the same runtime. With the decomposed approach the runtime is expected to be slightly lower than with the whole, with this depending essentially on the connectivity of the graph as described in section 4.1.2. However, this choice has not always been successful for all of the MI-FAP experiments performed (see Appendix A) whereas the results generally improve with the introduction of a further loop through the subsets.

7.5.1 Distribution of interference

The beneficial effect of a second loop is present within all the experiments presented in this chapter. Moreover, in some of the instances (for example the Bradford data sets in Tables A.14 and A.16), the decomposed approach is only effective at the end of this further loop, i.e. at the end of the first loop it still produces worse results than solving the problem as a whole when compared for the same number of evaluations (see Figure 7.16). This is essentially caused by a different distribution of the local interference inside each single subset at the end of each loop. An example of this is shown by the plots in Figure 7.22 for the Siemens1 benchmark with a geographical decomposition into two subsets. After the first loop the heuristic produces good solutions in the first subset, which, however, constraints the second subset leading to many violations. Finally, the second loop balances the interference between the subsets.

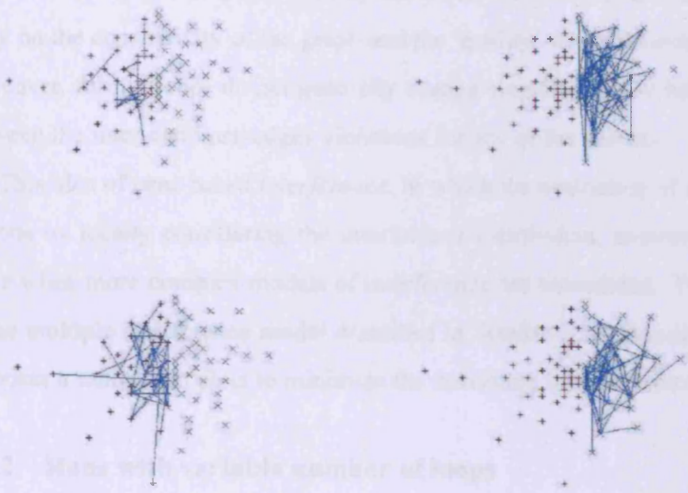


Figure 7.22: Local interference within each of the subsets after the first (top) and second (bottom) loop for COST259 Siemens1 with geographical decomposition into two subsets

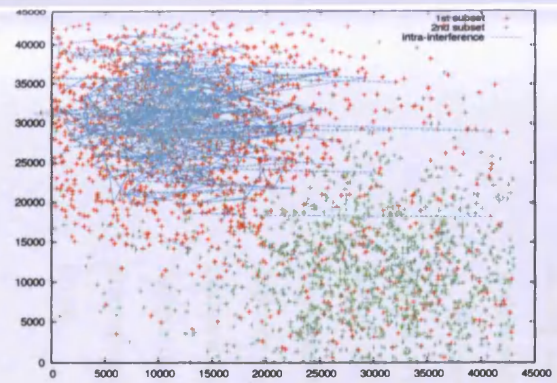
Investigating the local distribution of the interference is an alternative way of evaluating a given frequency assignment. Figures 7.23 shows, for the Cardiff University benchmark C_6 with a balanced GPFAP decomposition into two subsets, the interference produced in terms of violations of the constraints represented by the intra and inter-edges between subsets. We have here applied the decomposed assignment approach with the subsets considered in sequence for three loops through them (see Algorithm 3.1). During the first loop the first subset only produces violations of the constraints represented by its intra-edges, thus ignoring the inter-edges with the other subset. Then the partial assignment of the second subset completes the assignment but generally produces high interference values for both the inter and intra-edges violations. Subsequently, the second loop balances the interference between the subsets. Note that this reduces the violations of the inter-edges constraints between the two subsets, since eliminating this interference would extend the search space to the whole solution space, with ideally no loss of optimality between the solutions produced by the decomposed and the whole approach. However, this is only partially achieved by the redistribution loop and depends essentially on the connectivity of the graph and the 'quality' of the decomposition used. Moreover, further loops do not generally change significantly the balance reached between the inter and intra-edges violations for any of the subsets.

This idea of *area based interference*, in which the evaluation of an assignment is done by locally considering the interference distribution, assume more importance when more complex models of interference are considered. This is the case of the multiple interference model described in Section 2.2.4 (see also [40] which proposes a model that aims to minimize the maximum local interference).

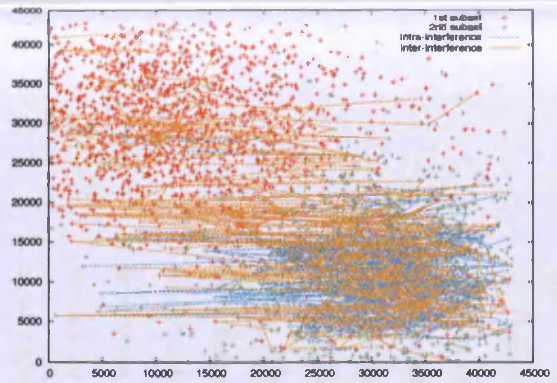
7.5.2 Runs with variable number of loops

If we consider the condition in Definition 4.1, given a number of total evaluations of a run with the decomposed assignment approach the introduction of a second loop through the subsets leads to a reduction of the number of evaluations computed for each subset. This can be done proportionally to the number of loops. For example, in our experiments, which consider only two loops (see Section 7.3), we have for each subset a number of evaluations explored equal to $eval_{i,0} = \frac{eval_{tot} * |V_i|}{2 * |V|}$.

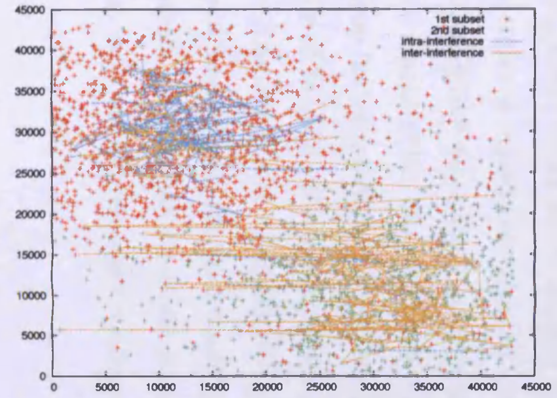
Note that, this will produce a different pair value, for the same fixed number of total evaluations, in the plots (cost, runtime) introduced in Section 4.1.3. For example in Figure 7.16 we have already shown two examples in which the desired number of evaluations $(2,000,000 * |V|)$ has been reached with either the whole approach, or a decomposition into two two subsets with one and two loops respectively.



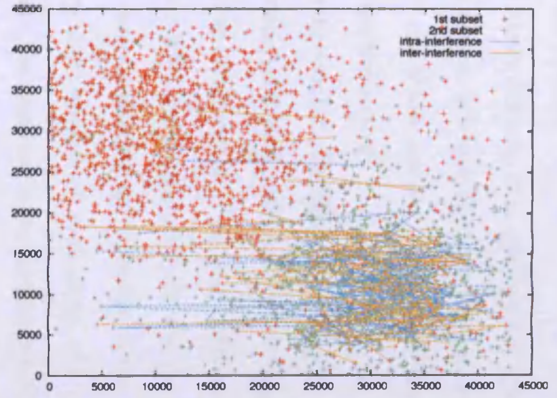
first loop - first subset



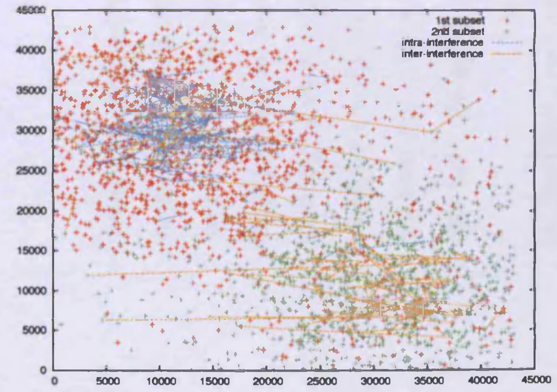
first loop - second subset



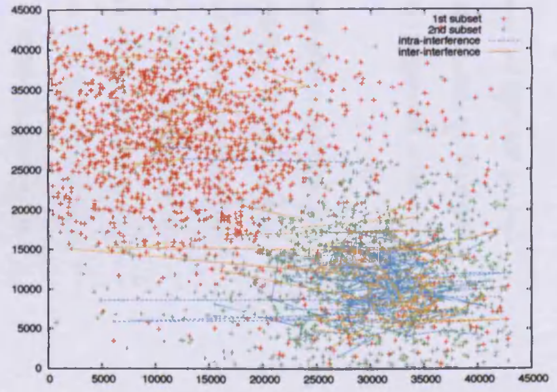
second loop - first subset



second loop - second subset



third loop - first subset



third loop - second subset

Figure 7.23: Intra and Inter-interference between subsets during the three loops of C_6 with GPFAP into two subsets.

For the same number of total evaluations we could add other loops and consequently reduce the number of evaluations per subsets, thus producing further {cost, runtime} pairs. However, this has not shown any significant improvement in the quality of the solutions for a number of preliminary experiments in which more than two loops were applied.

Figures 7.24 shows different runs of Siemens1 with a balanced GPFAP decomposition into two subsets in which the same number of total evaluations ($4,000,000* |V|$) is reached by the whole and the decomposed assignment approach with one, two and three loops respectively

Alternatively the number of evaluations computed for each subset can also vary during distinct loops (still maintaining fixed the desired number of total evaluations and satisfying Definition 4.1). In particular a number of tests have shown better performance when we considerably reduce the number of evaluations in the first subset (and correspondingly increase that of the others). Note that, since performed for a very short number of evaluations (for instance the 10% of the total in our experiments), the first loop assumes the significance of a preprocessing producing partial assignments which can be obtained either with the 'sequential' or 'independent' procedure in Algorithms 3.1 and 3.2 respectively.

We performed a GPFAP decompositions for a total of $4,000,000 * |V|$ evaluations. However, instead of this being computed by two loops of the same duration, the first loop has been further split into two parts. Firstly, we perform a preprocessing loop of $200,000 * |V|$ evaluations (solved either sequentially or independently),then a second loop of $1,800,000 * |V|$ completes the first half of the evaluations. Results Siemens2 and Bradford10 are shown in Table 7.14 and the decomposition approach results overall successful even towards the version without preprocessing (i.e. only one loop through the subsets of $2,000,000 * |V|$ evaluations).

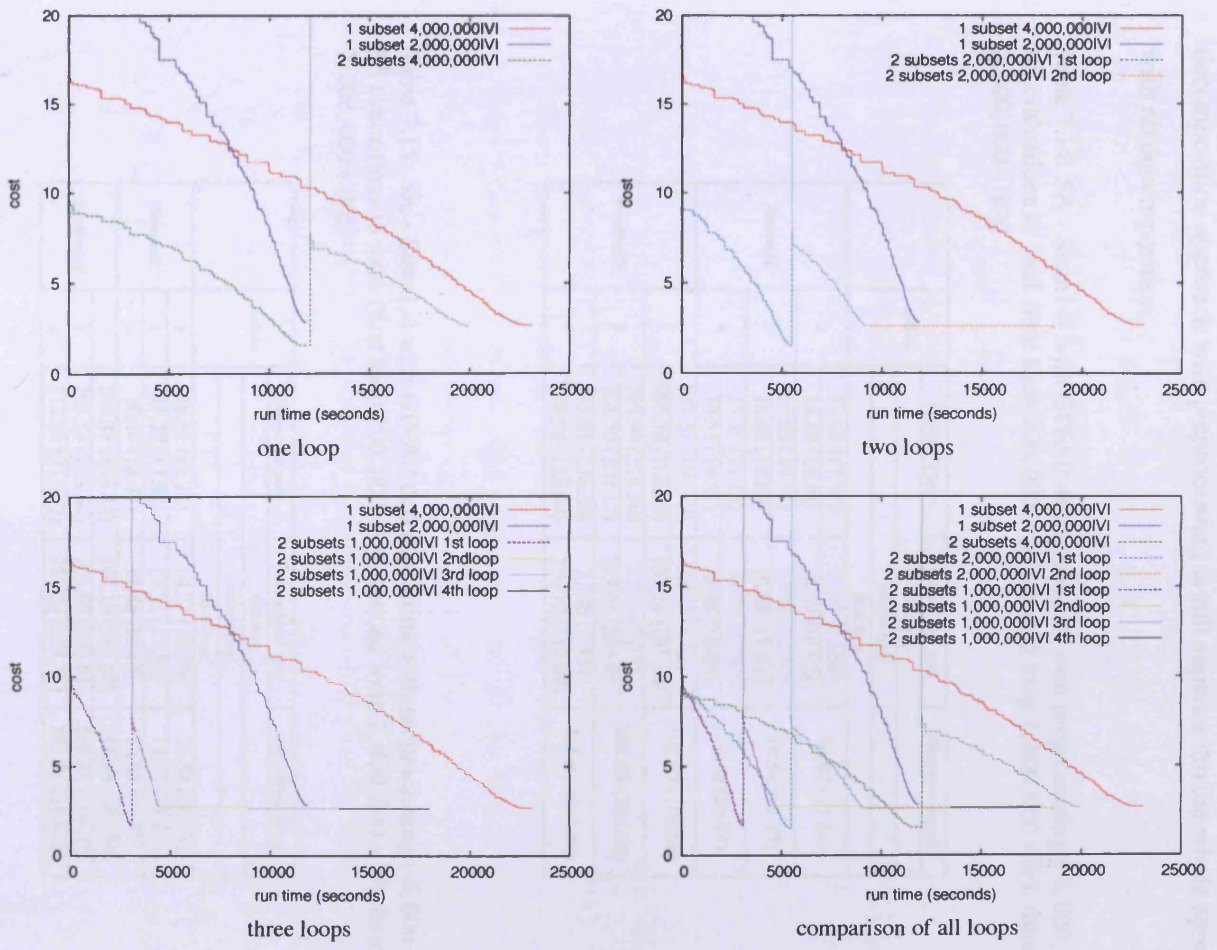


Figure 7.24: Cost-time plot for Siemens1 with balanced GPFAP decomposition into two subsets for $4,000,000 * |V|$ evaluations and different number of loops

Subsequently a further loop of the remaining $2,000,000 * |V|$ is computed through the subsets (see Table 7.15) after this further redistribution loop the normal decomposition approach with no preprocessing results more effective, although the decomposition approach with preprocessing is still superior tho the whole approach with no decomposition.

Table 7.14: SA - Siem1-4 with GPFAP decomposition with preprocessing ($4,000,000 * |V|$ evaluations in total (first loop $200,000 * |V|$, second loop $1,800,000 * |V|$, third loop $2,000,000 * |V|$))

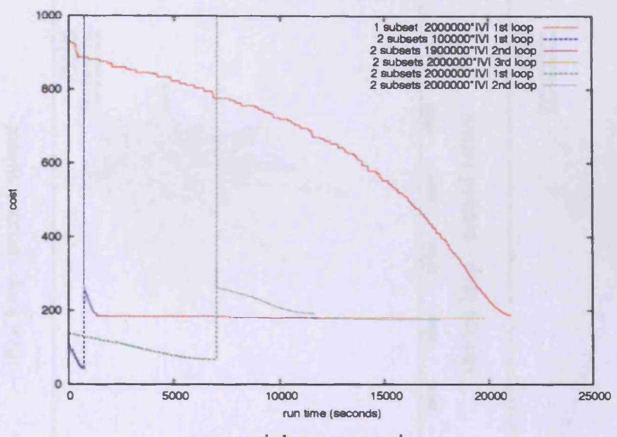
	noSub.	Sequential prep.	Independent prep	No preprocessing
		first loop		
		second loop		
		third loop		
Siemens2	2	17.60 (17.93)	5,050 (5,150)	16.97 (17.04)
		16.38 (16.41)	16.49 (16.92)	
	3	19.85 (20.51)	5,202 (5,234)	19.54 (19.77)
		18.08 (18.55)	18.89 (19.02)	
	4	20.95 (21.12)	5,472 (5,760)	20.56 (20.87)
		19.35 (19.83)	**18.56 (18.60)**	
Bradford10	2	197.62 (197.98)	2,015 (2,022)	192.37 (192.98)
		187.74 (188.22)	191.26 (191.54)	
	3	225.90 (226.30)	2,023 (2,027)	**203.69 (204.60)**
		206.70 (207.17)	205.01 (205.86)	
	4	233.01 (236.56)	2,348 (2,354)	**211.81 (212.94)**
		228.28 (228.84)	216.52 (217.09)	

Table 7.15: SA - Siem1-4 with GPFAP decomposition with preprocessing ($4,000,000 * |V|$ evaluations in total (first loop $200,000 * |V|$, second loop $1,800,000 * |V|$, third loop $2,000,000 * |V|$))

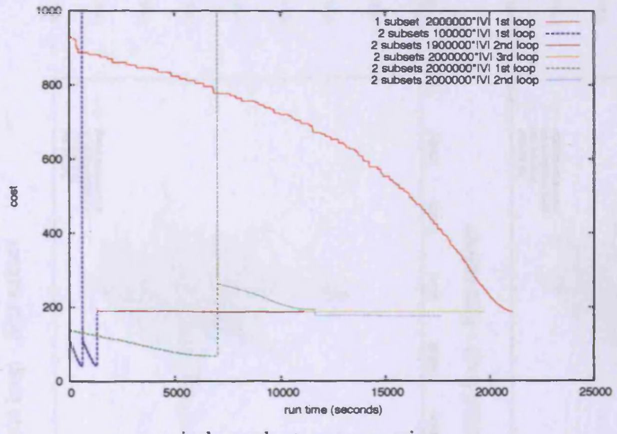
	noSub.	Sequential prep.	Independent prep	No preprocessing
		first loop		
		second loop		
		third loop		
Siemens2	2	**16.11 (16.13)**	16.24 (16.35)	16.34 (16.73)
	3	**17.27 (17.52)**	17.51 (17.91)	17.41 (17.60)
	4	18.38 (18.47)	**17.88 (18.19)**	18.02 (18.27)
Bradford10	2	186.31 (186.42)	190.19 (189.69)	**185.05 (185.39)**
	3	203.53 (203.64)	201.95 (202.01)	**188.38 (189.51)**
	4	212.81 (215.27)	213.34 (214.57)	**193.51 (194.43)**

At the end of the first loop both ‘sequential’ and ‘independent’ preprocessing approaches improve the results obtained in Table 7.8 ($2,000,000 * |V|$ evaluations per loop). Hence the separation of the first loop into a preprocessing phase, which produces a quick approximation of an optimal assignment, and a further loop (which actually redistributes the interference among the subsets), performs better than a single loop performed sequentially through the subsets. However, this mainly happens for a decomposition into two subsets only, with 'sequential' preprocessing superior to 'independent'. Nevertheless, the subsequent final loop (the third loop in our experiments) does not further improve the results which are in general only comparable with those in Table 7.8 at the end of the $4,000,000 * |V|$ evaluations. Figure 7.25 shows an example of two single runs of Bradford10 with a decomposition into two subsets for a total of $4,000,000 * |V|$ evaluations, in which the preprocessing approach is compared with the same decomposition run for two loops of equal duration (see also Table 7.8). When the preprocessing approach is applied a total of thee lopps are performed of $100,000 * |V|$, $1,900,000 * |V|$, and $3,000,000 * |V|$ evaluations respectively whereas with the normal decomposition method used throughout this thesis only two loops of $4,000,000 * |V|$ evaluations each are computed.

Figures 7.26 and 7.27 show the distribution of the interference within the three loops for Siemens2 solved by the decomposed assignment approach adopting the 'independent' preprocessing. We have considered GPFAP and geographical decomposition into two subsets. The plots shows that the first loop involves only the intra-edge interference within each of subsets, whereas the second loop has the effect of redistributing the interference balancing the inter and intra-edges violations. This effect is more important for 'good' decompositions like GPFAP rather than the worse performing geographical partitioning. Finally, the subsequent third loop does not change significantly the relative distribution of the local interference for both of the decomposition methods used. Nevertheless, since the connectivity of this specific data set is very high the interference in terms of number of inter and intra-edges violations appears high for both of the examples tested.



sequential preprocessing



independent preprocessing

Figure 7.25: Cost-time plot for Bradford10 with balanced GPFAP decomposition into two subsets for $4,000,000 * |V|$ evaluations with preprocessing approach.

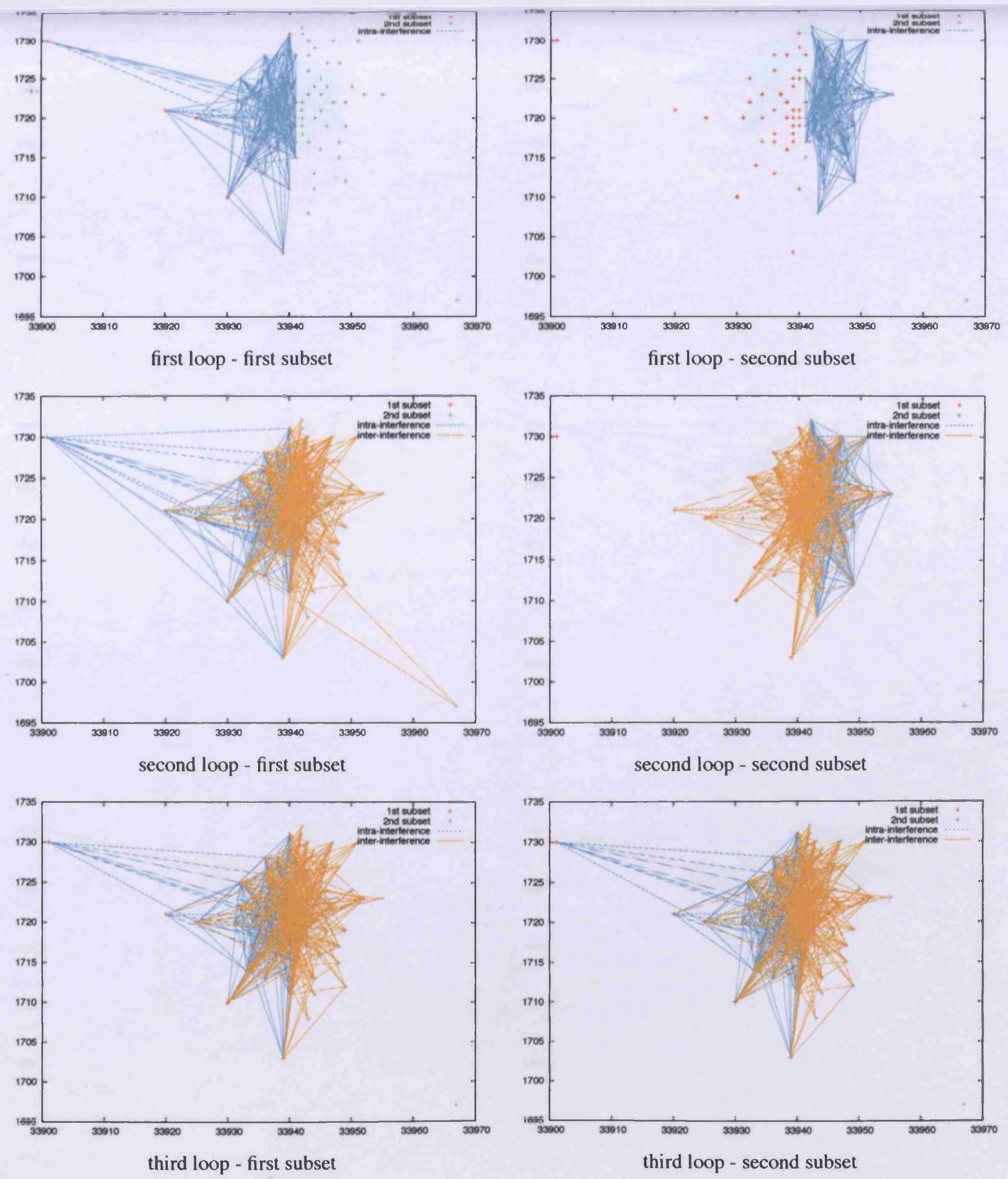
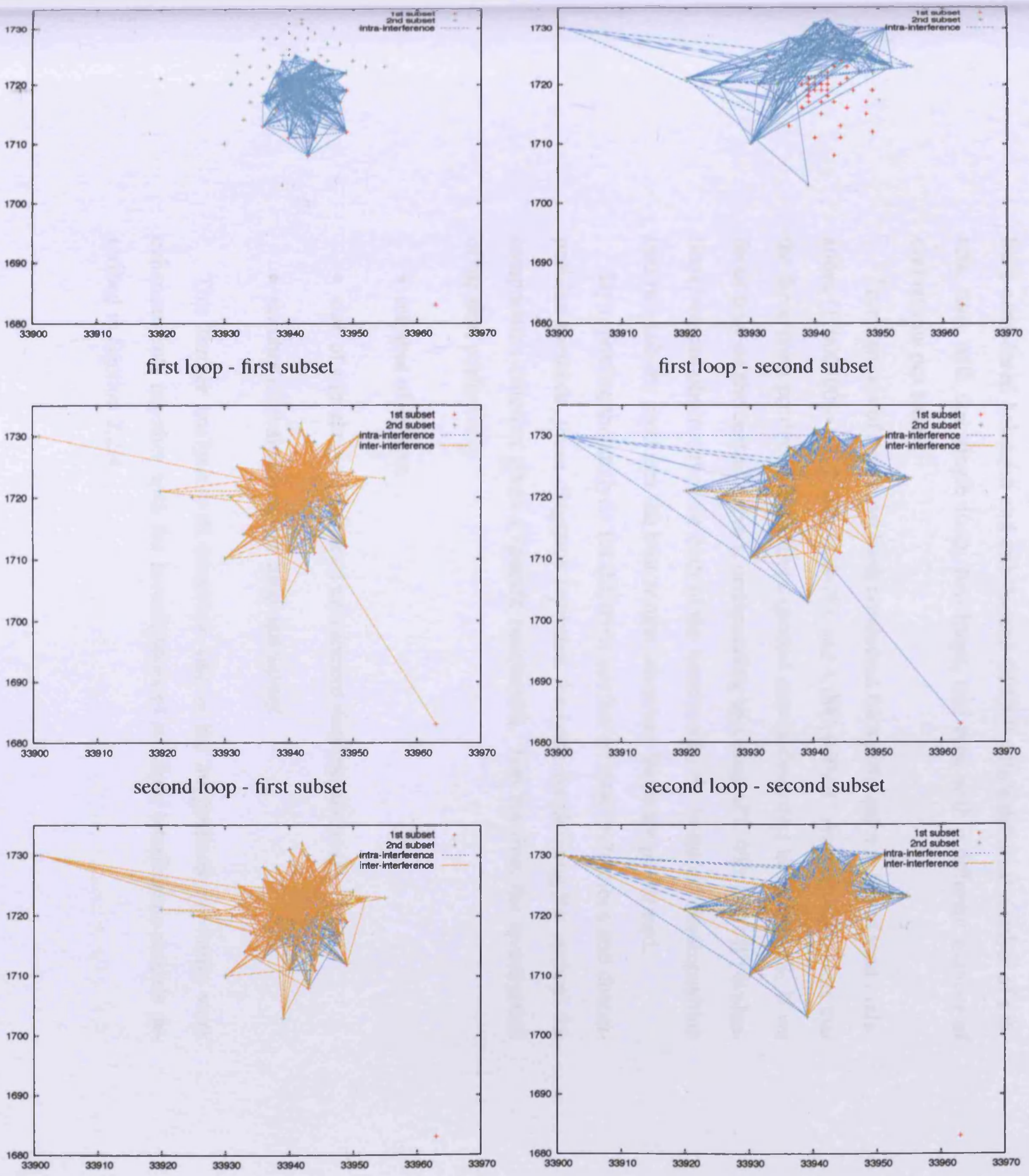


Figure 7.26: Intra and Inter-interference between subsets during the three loops of Siemens2 with geographical decomposition into two subsets.



first loop - first subset

first loop - second subset

second loop - first subset

second loop - second subset

third loop - first subset

third loop - second subset

Figure 7.27: Intra and Inter-interference between subsets during the three loops of Siemens2 with GPFAP into two subsets.

7.5.3 Cost-runtime trade-off

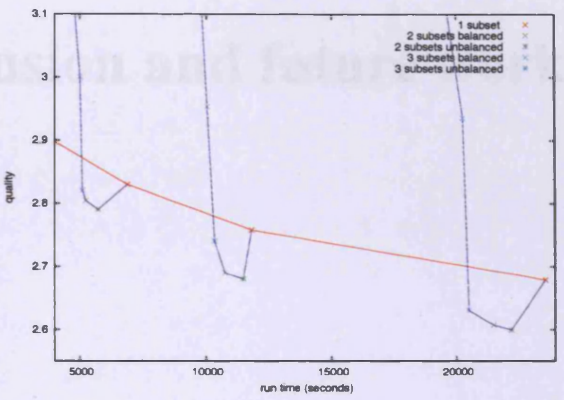
Figure 7.28 shows the cost-runtime curve for Siemens1 and Bradford10 respectively for a number of runs with and without the decomposed assignment approach (with the points corresponding to the whole approach joined with a red line). We have considered balanced and unbalanced GPFAP with a different number of subsets, runs with one single loop, two loops, and runs with a different number of evaluations per loop.

Three groups of runs have been conducted for a different number of total evaluations ($1,000,000 * |V|$, $2,000,000 * |V|$, and $4,000,000 * |V|$ respectively). Note that the three plots produced recall the expected curves described in Figure 4.3. If we focus only on the central curve (corresponding to a total of $2,000,000 * |V|$ evaluations) we can observe that for both of the benchmarks the balanced decomposition into two subsets produces the best results when two loops are performed.

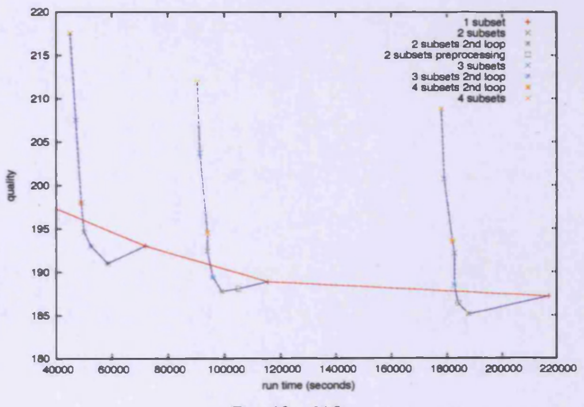
By repeating the analysis for different number of total evaluations and decomposition methods, these diagrams represent the basis for defining the optimal decomposition criterion given a specific benchmark. This involves the investigation of the best performing:

- number of subsets
- size of subsets (i.e. balanced unbalanced decompositions)
- number of evaluations per loop and subset

This further analysis will constitute one of the suggestions for future work enhancements together with the investigation of multiple interference models described in Section 2.2.4.



Siemens1



Bradford10

Figure 7.28: Trade-off between cost and runtime for Siemens1 and Bradford10

Chapter 8

Conclusion and future work

8.1 Conclusion

This thesis proposes a decomposed meta-heuristic approach to solve the FAP in its different formulations (minimum span (MS-FAP), fixed spectrum (FS-FAP) and minimum interference frequency assignment (MI-FAP)). The FAP can be solved with exact methods only for small networks with an order of few hundreds transmitters whereas for larger problems is usually solved by heuristic algorithms. However, although meta-heuristics produce good results on some of the benchmarks available, they often perform on specific data sets for the particular type of FAP considered and highly specialised algorithms tend to perform best. In addition, standard implementations of meta-heuristics may require considerable runtimes to produce good quality results whenever a problem is very large or complex.

This thesis investigates the application of problem decomposition techniques as an effective approach in order to deal with large networks. Although a few previous works have applied problem decomposition in combination with exact methods these techniques have very rarely been used in combination with meta-heuristics. Furthermore, in these cases the decomposition technique has been mainly applied either after or incorporated into a heuristic procedure in order to optimizing locally the solution produced (often using again exact procedures).

In this thesis the decomposition strategy has been extended to a larger scale which aims to simplify a complicated problem by decomposing it into separate subproblems (obtained by removing some of the constraints between them) rather than increasing the complexity of the meta-heuristic which solves the problem as a whole. A similar approach has only been proposed in the literature for the easiest formulations of the FAP (MS-FAP) and for a specific set of benchmarks and decomposition methods.

Our proposed decomposed assignment approach is based on an initial partition of the interference graph representing the network into two or more subgraphs. A meta-heuristic procedure is then applied to each of the subsets in turn to produce a sequence of partial solutions. Subsets can be solved either sequentially or independently, that is when the current subset is considered the algorithm keeps the assignment of the transmitters in the previously assigned subsets fixed respecting the constraint violations between them. Finally, the partial solutions are recomposed to give a complete assignment of the original problem.

This thesis constitutes the first attempt to constructively investigate the effectiveness of decomposition approaches for the FAPs solved in combination with meta-heuristics. We propose a range of decomposition methods to produce a partitioning of the interference graph representing the network. These include the generalization of methods previously used for the FAP, such as clique detection and partitions based on generalized degree, together with novel applications and modifications of existing graph partitioning and clustering methods.

The idea of decomposing the hardest FAP benchmarks into a partition of subproblems is particularly important for standard meta-heuristics, which are not otherwise capable of producing satisfactory performance for the more complex types of FAP such as the MI-FAP. A number of decomposition algorithms have been applied to a standard simulated annealing and a genetic algorithm with two different representations (direct and order-based). Both algorithms have been applied to the FAP in its different models. However, more attention has been given to the more complex and useful formulation of the problem, the MI-FAP. From an analysis of the results we can draw the following conclusions.

The order-based GA obtained good results with the MS-FAP and simple instances of the FS-FAP, while for harder FS-FAP benchmarks and the MI-FAP the use of a direct representation produces better results. Furthermore, for the MI-FAP the use of a multi-objective approach outperforms the single-objective, especially in terms of diversification of the partial solutions in the population. However, simulated annealing performs better than the GA in all of the MI-FAP experiments conducted.

For the majority of the instances the decomposed assignment with a decomposition into two to four subsets performs better than the whole approach with an evaluation on a pure quality basis. Results show that different partitioning methods are preferable to solve the different types of FAP. For the MS-FAP the more effective methods are those based on solving the 'hardest' part of the problem first (clique and generalized-degree), thus confirming the results already obtained in previously published works, whereas for the harder MI-FAP (and its generalization as FS-FAP) the methods based on minimizing the interconnections between different subsets give the best performance (i.e. graph partitioning and clustering).

Furthermore, when the problem is solved by the GA the order-based representation is more effective in solving the MS-FAP while for the FS/MI-FAP the direct representation appears superior. This may be partially explained by the fact that for the MS-FAP removing some of the vertices still allows the core part of an instance to produce a span which is close to that of the whole data sets. On the contrary, when the cost is defined in terms of the total sum of the constraint violations (as for the FS-FAP), removing edges unavoidably implies a considerable decrease in the partial costs produced by a subproblem.

With the order-based representation, the sequential assignment used considers only the vertices already assigned in a given ordering in the current subset, thus ignoring a fraction of transmitters in each subset (those next to come in the given ordering of the subsets vertices). We can then see these vertices as they were removed from the subset. For the reason above, this may penalize this type of representation when used to solve the FS-FAP whereas for the MS-FAP the cost (span) produced for each of the subsets during an assignment is less affected from this removal of vertices with a beneficial effect on the final span returned by the decomposition technique.

Some groups of benchmarks perform better with specific decomposition methods. This is, for example, the case of the Cardiff University data sets which, since they present a location of the transmitters distributed into 'towns', clearly produce better results with the graph partitioning method. For other group of benchmarks, such as the Bradford data sets for the MI-FAP, the decomposition approach is not immediately successful on a pure quality basis and requires the introduction of a further reassignment loop over the subsets. This has the consequence of improving considerably the quality of the results as well as balancing the distribution of the interference in the network.

For some benchmarks, which present either a very high graph density or a peculiar structure (as predicted for random graphs), the performance of the decomposed assigned approach is not satisfactory with an evaluation purely based on quality. However, if we look at the the trade-off between quality and runtime during a single run of the meta-heuristic the decomposition approach appears able to produce acceptable approximations of the optimal solution in a shorter time than the whole approach (when both the approaches are run and compared for a same number of solutions explored) as described in more detail in Section 4.1.3.

In particular, very good results in this sense are obtained with the largest set of benchmarks used, which represent more realistic instances of wireless networks. Here, the GPFAP is able to effectively solve MI-FAP instances without the need of any further knowledge about the network (e.g. geographical information about transmitter locations). This result constitutes the main contribution of this thesis which then proposes the decomposed assignment approach as an effective technique (and in some cases the only one possible), to solve larger practical data sets using meta-heuristics.

Beside this main result, our work has shown that the decomposed approach constitutes a valid technique to improve the performance of standard meta-heuristics in a way which is actually algorithm independent. This is evident for the COST-259 MI-FAP results for which not only has this approach improved the previously published results produced by the standard implementation of SA but it has also allowed the GA to be more competitive for this problem, whereas no published results are yet known for these benchmarks for this category of evolutionary meta-heuristics. Finally, the decomposed assignment approach (when effective) allows the use of standard meta-heuristics independently on the data set used, thus avoiding the use of algorithms specifically designed for a particular class of benchmarks.

8.2 Future work

This final section suggests two main future directions toward which the research proposed in this thesis can be extended.

Firstly, the proposed procedure which, given a test problem and a decomposition method, deduces the optimal number of subsets in the decomposition could be improved and better specified. We have here suggested the use of cost-time curves which plot different pairs of cost time values corresponding to an assigned total number of solutions explored by the meta-heuristic, for both the whole and the decomposed approach, thus expected to produce roughly the same runtime. Different pair values can be produced varying the number of the subsets (and when applicable their size) as well as the number of loops performed over them. However, this technique presents the drawback of depending strictly on the specific instance considered and alternative procedure may be therefore considered. In addition, either further decomposition methods or variations of those already proposed in this thesis may be worthy of being investigated.

A second, more important suggestion for future enhancements consists of extending the application of the problem decomposition approach for the FAP to more complex interference models than the binary considered here. This will involve the adoption of a multiple interference model, which abandons the idea of considering separately pairs of transmitters and takes instead into account the cumulative effect of the interfering signals received by a transmitter when all of the other geographically close base stations transmit simultaneously. This approach may also require further formulations of the FAP beside the minimization of the global interference in the network, such as minimizing the maximal interference experienced locally in the network by a transmitter (see also the concept of area based interference introduced in the final part of this thesis).

Appendix A

Extended results

A.1 Extended results for MS-FAP

We present in this section of the appendix the extended results obtained for the MS-FAP by the order-based GA with the decomposed assignment approach. Tables A.2, A.3, and A.1 show the comparative results expressed in percentage obtained for the MS-FAP benchmarks with the decomposition methods outlined in Table 5.1.

Table A.1: Comparative results in % for MS-FAP. Best span for for R_1, and R_2 solved by the order-based GA with decomposition ($500,000 * |V|$ evaluations)

		GPFAP		Gen. Degree		Cliques		Whole
				R_1				
GPFAP	2	–	–	+4	+3	+2	+1	+3
	4	–	–	+6	+5	+3	+2	+4
Gen Degree	2	-4	-6	–	–	-3	-3	-1
	4	-3	-5	–	–	-2	-2	0
Cliques	2	-2	-3	+3	+2	–	–	+2
	4	-1	-2	+3	+2	–	–	+2
Whole	1	-3	-4	+1	0	-2	-2	–
				R_2				
GPFAP	2	–	–	+4	+4	+3	+3	+4
	4	–	–	+5	+5	+4	+4	+5
Gen Degree	2	-4	-5	–	–	-1	-1	0
	4	-4	-5	–	–	-1	-1	0
Cliques	2	-3	-4	+1	+1	–	–	+1
	4	-3	-4	+1	+1	–	–	+1
Whole	1	-4	-5	0	0	-1	-1	–

Table A.2: Comparative results in % for MS-FAP. Best span for for P_1, P_2, C_1, C_2, and C_3 solved by the order-based GA with decomposition ($500,000 * |V|$ evaluations)

		Random		Geog.		GPFAP		Gen. Degree		Cliques		Whole
		2	4	2	4	2	4	2	4	2	4	1
						P_1						
Random	2	–	–	–	–	+8	+6	+8	+8	+8	+6	+8
	4	–	–	–	–	+8	+6	+8	+8	+8	+6	+8
Geog	2	–	–	–	–	–	–	–	–	–	–	–
	4	–	–	–	–	–	–	–	–	–	–	–
GPFAP	2	-8	-8	–	–	–	–	0	0	0	-2	0
	4	-6	-6	–	–	–	–	+2	+2	+2	0	+2
Gen Degree	2	-8	-8	–	–	0	-2	–	–	0	-2	0
	4	-8	-8	–	–	0	-2	–	–	0	-2	0
Cliques	2	-8	-8	–	–	0	-2	0	0	–	–	0
	4	-6	-6	–	–	+2	0	+2	+2	–	–	+2
Whole	1	-8	-8	–	–	0	-2	0	0	0	-2	–
						P_2						
Random	2	–	–	–	–	+12	+12	+12	+12	+11	+11	+12
	4	–	–	–	–	+27	+27	+27	+27	+26	+26	+27
Geog	2	–	–	–	–	–	–	–	–	–	–	–
	4	–	–	–	–	–	–	–	–	–	–	–
GPFAP	2	-12	-27	–	–	–	–	0	0	-1	-1	0
	4	-12	-27	–	–	–	–	0	0	-1	-1	0
Gen Degree	2	-12	-27	–	–	0	0	–	–	-1	-1	0
	4	-12	-27	–	–	0	0	–	–	-1	-1	0
Cliques	2	-11	-26	–	–	+1	+1	+1	+1	–	–	+1
	4	-11	-26	–	–	+1	+1	+1	+1	–	–	+1
Whole	1	-12	-27	–	–	0	0	0	0	-1	-1	–
						C_1						
Random	2	–	–	+6	+9	+20	+16	+25	+27	+22	+19	+23
	4	–	–	+13	+15	+28	+23	+33	+34	+29	+26	+31
Geog	2	-6	-13	–	–	+13	+9	+18	+19	+15	-12	+16
	4	-9	-15	–	–	+11	+7	+15	+16	+12	+10	+14
GPFAP	2	-20	-28	-13	-11	–	–	+4	+5	+1	-1	+2
	4	-16	-23	-9	-7	–	–	+8	+9	+0.5	+0.2	+0.6
Gen Degree	2	-25	-33	-18	-15	-4	-8	–	–	-3	-5	-1
	4	-27	-34	-19	-16	-0.5	-0.9	–	–	-4	-6	-3
Cliques	2	-22	-29	-15	-12	-0.1	-0.5	+3	+4	–	–	+1
	4	-19	-26	-12	-10	+0.1	-0.2	+5	+6	–	–	+4
Whole	1	-23	-31	-16	-14	-0.2	-0.6	+1	+3	-1	-4	–
						C_2						
Random	2	–	–	+1	0	+5	+8	+15	+16	+10	+9	+15
	4	–	–	+5	+4	+9	+11	+19	+21	+14	+13	+19
Geog	2	-1	-5	–	–	+4	+6	+14	+15	+9	+8	+14
	4	0	-4	–	–	+5	+8	+15	+16	+10	+9	+15
GPFAP	2	-5	-9	-4	-5	–	–	+9	+10	+5	+4	+9
	4	-8	-11	-6	-8	–	–	+0.7	+8	+2	+1	+7
Gen Degree	2	-15	-19	-14	-15	-9	-7	–	–	-4	-5	0
	4	-16	-21	-15	-16	-10	-8	–	–	-5	-7	-1
Cliques	2	-10	-14	-9	-10	-5	-2	+4	+5	–	–	+4
	4	-9	-13	-8	-9	-4	-1	+5	+7	–	–	+5
Whole	1	-15	-19	-14	-15	-9	-7	0	+1	-4	-5	–
						C_3						
Random	2	–	–	+8	+6	+12	+13	+20	+22	+13	+16	+12
	4	–	–	+10	+7	+15	+14	+22	+23	+14	+12	+22
Geog	2	-8	-10	–	–	+5	+4	+11	+12	+4	+6	+11
	4	-6	-7	–	–	+8	+6	+14	+15	+6	+9	+14
GPFAP	2	-12	-15	-5	-5	–	–	+5	+7	-1	+1	+5
	4	-13	-14	-4	-6	–	–	+9	+8	0	+3	+9
Gen Degree	2	-20	-22	-11	-14	-5	-9	–	–	-6	-4	0
	4	-22	-23	-12	-15	-7	-8	–	–	-8	-5	-1
Cliques	2	-13	-14	-4	-6	+1	0	+6	+8	–	–	+6
	4	-16	-12	-6	-9	[illegible]	-3	+4	+5	–	–	+4
Whole	1	-12	-22	-11	-14	-5	-9	0	+1	-6	-4	–

Table A.3: Comparative results in % for MS-FAP. Best span for for C_4, G_1, G_2, and G_3 solved by the order-based GA with decomposition ($500,000 * |V|$ evaluations)

		Random		Geog.		GPFAP		Gen. Degree		Cliques		Whole
C_4												
Random	2	–	–	+38	+34	+42	+32	+34	+38	+34	+36	+31
	4	–	–	+49	+45	+54	+43	+45	+49	+45	+43	+41
Geog	2	.38	.49	–	–	+3	.4	.3	0	.3	.1	.6
	4	.34	.45	–	–	+6	.1	0	+3	0	+1	.3
GPFAP	2	.42	.54	.3	.6	–	–	.6	.3	.6	.4	.7
	4	.32	.43	+4	+1	–	–	+1	+4	+1	+3	.1
Gen Degree	2	.34	.45	+3	0	+6	.1	–	–	0	+1	.3
	4	.38	.49	0	.3	+3	.4	–	–	.3	.1	.6
Cliques	2	.34	.45	+3	0	+6	.1	0	+3	–	–	.3
	4	.36	.43	+1	.1	+4	.3	.1	+1	–	–	.4
Whole	2	.31	.41	+6	+3	+7	+1	+3	+6	+3	+4	–
G_1												
Random	2	–	–	–	–	+3	0	+3	+10	+3	+2	+4
	4	–	–	–	–	+8	+2	+8	+12	+8	+4	+6
GPFAP	2	-3	-8	–	–	–	–	0	+4	0	-4	-2
	4	0	-2	–	–	–	–	+6	+10	+6	+2	+4
Gen Degree	2	-3	-8	–	–	0	-6	–	–	0	-4	-2
	4	-10	-12	–	–	-4	-10	–	–	-4	-8	-6
Cliques	2	-3	-8	–	–	0	-6	0	+4	–	–	-2
	4	-2	-4	–	–	+4	-2	+4	+8	–	–	+2
Whole	1	-4	-6	–	–	+2	-4	+2	+6	+2	-2	–
G_2												
Random	2	–	–	–	–	+14	+22	+20	+22	+20	+20	+20
	4	–	–	–	–	+15	+24	+22	+24	+22	+22	+22
GPFAP	2	-14	-15	–	–	–	–	+5	+7	+5	+5	+5
	4	-22	-24	–	–	–	–	-2	0	-2	-2	-2
Gen Degree	2	-20	-22	–	–	-5	+2	–	–	0	0	0
	4	-22	-24	–	–	-7	0	–	–	-2	-2	-2
Cliques	2	-20	-22	–	–	-5	+2	0	+2	–	–	0
	4	-20	-22	–	–	-5	+2	0	+2	–	–	0
Whole	1	-20	-22	–	–	-5	+2	0	+2	0	0	–
G_3												
Random	2	–	–	–	–	+7	+5	+8	+5	+8	+8	+8
	4	–	–	–	–	+11	+8	+12	+8	+12	+12	+12
GPFAP	2	-7	-11	–	–	–	–	+1	0	+1	+1	+1
	4	-5	-8	–	–	–	–	+4	+3	+4	+4	+4
Gen Degree	2	-8	-12	–	–	-1	-4	–	–	0	0	0
	4	-5	-8	–	–	0	-3	–	–	+1	+1	+1
Cliques	2	-8	-12	–	–	-1	-4	0	-1	–	–	0
	4	-8	-12	–	–	-1	-4	0	-1	–	–	0
Whole	1	-8	-12	–	–	-1	-4	0	-1	0	0	–

A.2 Extended results for FS-FAP

We present in this section of the appendix the extended results obtained for the FS-FAP by the order-based and direct GA with the decomposed assignment approach. Tables A.5, A.6 and A.4 show the comparative results expressed in percentage for the FS-FAP benchmarks with the decomposition methods outlined in Table 6.1 and 6.5. The cost reported in the table is the one defined in Problem 1.2 for this type of FAP.

Table A.4: Comparative results in % for FS-FAP. Best span for for R_1 and R_2 solved by the direct GA with decomposition $(500,000 * |V|$ evaluations)

		GPFAP		Gen. Degree		Cliques		Whole
		2	4	2	4	2	4	1
				R_1				
GPFAP	2	–	–	-3	-9	-2	-8	+2
	4	–	–	+3	-2	+4	-2	+8
Gen Degree	2	+3	-3	–	–	+1	-4	+5
	4	+9	+2	–	–	+6	+1	+11
Cliques	2	+2	-4	-1	-6	–	–	+4
	4	+8	+2	+4	-1	–	–	+9
Whole	2	-2	-8	-5	-11	-4	-9	–
				R_2				
GPFAP	2	–	–	-5	-18	+6	+4	+6
	4	–	–	+2	-10	+13	+11	+13
Gen Degree	2	+5	-2	–	–	+11	+9	+11
	4	+18	+10	–	–	+26	+24	+26
Cliques	2	-6	-13	-11	-26	–	–	0
	4	-4	-11	-9	-24	–	–	+1
Whole	2	-6	-13	-11	-26	0	-1	–

Table A.5: Comparative results in % for FS-FAP. Best cost for for C_1, C_2, C_3, and C_4 solved by the order-based GA with decomposition ($500,000 * |V|$ evaluations)

		Random		Geog.		GPFAP		Gen. Degree		Cliques		Whole
		2	4	2	4	2	4	2	4	2	4	1
C_1												
Random	2	–	–	+44	+31	+57	+53	+52	+57	+47	+50	+53
	4	–	–	+44	+31	+57	+53	+52	+57	+47	+50	+53
Geog	2	-44	-44	–	–	+9	+6	+5	+8	+2	+4	+6
	4	-31	-31	–	–	+20	+17	+16	+20	+12	+15	+17
GPFAP	2	-57	-57	-9	-20	–	–	-3	0	-6	-4	-2
	4	-53	-53	-6	-17	–	–	-1	+2	-4	-2	0
Gen Degree	2	-52	-52	-5	-16	+3	+1	–	–	-3	-1	+1
	4	-57	-57	-8	-20	0	-2	–	–	-7	-4	-2
Cliques	2	-47	-47	-2	-12	+6	+4	+3	+7	–	–	+4
	4	-50	-50	-4	-15	+4	+2	+1	+4	–	–	+2
Whole	1	-53	-53	-6	-17	+2	0	-1	+2	-4	-2	–
C_2												
Random	2	–	–	+15	-3	+95	+63	+85	+1.00	+50	+56	+59
	4	–	–	+26	+8	+1.15	+79	+1.04	+1.20	+63	+72	+76
Geog	2	-15	-26	–	–	+70	+42	+62	+37	+31	+36	+39
	4	+3	-8	–	–	+1.00	+66	+90	+1.05	+54	+60	+63
GPFAP	2	-95	-1.15	-70	-1.00	–	–	-5	+2	-30	-25	-23
	4	-63	-79	-42	-66	–	–	+14	+23	-8	-4	-2
Gen Degree	2	-85	-1.04	-62	-90	+5	-14	–	–	-24	-19	-16
	4	-1.00	-1.20	-37	-1.05	-2	-23	–	–	-33	-28	-26
Cliques	2	-50	-63	-31	-54	+30	+8	+24	+33	–	–	+6
	4	-56	-72	-36	-60	+25	+4	+19	+28	–	–	+2
Whole	2	-59	-76	-39	-63	+23	+2	+16	+26	-6	-2	–
C_3												
Random	2	–	–	+31	+12	+37	+29	+31	+30	+20	+10	+27
	4	–	–	+29	+11	+35	+27	+29	+28	+19	+9	+26
Geog	2	-31	-29	–	–	+5	-1	0	-1	-9	-19	-3
	4	-12	-11	–	–	+22	+15	+17	+16	+10	-2	+13
GPFAP	2	-37	-35	-5	-22	–	–	-5	-5	-14	-24	-8
	4	-29	-27	+1	-15	–	–	+1	+1	-7	-17	-3
Gen Degree	2	-31	-29	0	-17	+5	-1	–	–	-9	-19	-3
	4	-30	-28	+1	-16	+5	-1	–	–	-8	-18	-2
Cliques	2	-20	-19	+9	-10	+14	+7	+9	+8	–	–	+10
	4	-10	-9	+19	+2	+24	+17	+19	+18	–	–	+16
Whole	2	-27	-26	+3	-13	+8	+3	+3	+2	-10	-16	–
C_4												
Random	2	–	–	+20	+19	+21	+21	+18	+11	+13	+14	+14
	4	–	–	+25	+24	+26	+27	+23	+16	+18	+19	+19
Geog	2	-20	-25	–	–	+1	+1	-1	-7	-6	-5	-5
	4	-19	-24	–	–	+2	+2	0	-6	-5	-3	-3
GPFAP	2	-21	-26	-1	-2	–	–	-2	-8	-7	-6	-6
	4	-21	-27	-1	-2	–	–	-3	-9	-7	-6	-6
Gen Degree	2	-18	-23	+1	0	+2	+3	–	–	-4	-3	-03
	4	-11	-16	+7	+6	+8	+9	–	–	+2	+3	-6
Cliques	2	-13	-18	+6	+5	+7	+7	+4	-2	–	–	+1
	4	-14	-19	+5	+3	+6	+6	+3	-3	–	–	0
Whole	1	-14	-19	+5	+3	+6	+6	+3	+6	-1	0	–

Table A.6: Comparative results in % for FS-FAP. Best span for for G_1 and G_2 solved by the order-based GA with decomposition ($500,000 * |V|$ evaluations)

		Random		Geog.		GPFAP		Gen. Degree		Cliques		Whole
		2	4	2	4	2	4	2	4	2	4	1
						G_1						
Random	2	–	–	–	–	+34	+37	+24	+28	+27	+30	+29
	4	–	–	–	–	+34	+38	+29	+25	+28	+31	+30
GPFAP	2	-34	-34	–	–	–	–	-4	-8	-5	-2	-4
	4	-37	-38	–	–	–	–	-7	-10	-7	-4	-6
Gen Degree	2	-24	-29	–	–	+4	+7	–	–	-1	+2	+1
	4	-28	-25	–	–	+8	+10	–	–	+3	+5	+4
Cliques	2	-27	-28	–	–	+5	+7	+1	+3	–	–	+1
	4	-30	-31	–	–	+2	+4	+2	+5	–	–	-1
Whole	2	-29	-30	–	–	+4	+6	-1	-4	-1	+1	–
						G_2						
Random	2	–	–	–	–	+27	+21	+24	+22	+26	+21	+20
	4	–	–	–	–	+33	+26	+30	+28	+33	+27	+26
GPFAP	2	-27	-33	–	–	–	–	-2	-4	0	-5	-5
	4	-21	-26	–	–	–	–	+3	+1	+5	0	-1
Gen Degree	2	-24	-30	–	–	+2	-3	–	–	+2	-3	-3
	4	-22	-28	–	–	+4	-1	–	–	+4	-1	-2
Cliques	2	-26	-33	–	–	0	-5	-2	-4	–	–	-6
	4	-21	-27	–	–	+5	0	+3	+1	–	–	-1
Whole	2	-20	-26	–	–	+5	+1	+3	+2	+6	+1	–

A.3 Extended results for MI-FAP

We present in this section of the appendix the extended results obtained for the MI-FAP by SA and the GA for a variable number of evaluations. Table A.7, A.8, and A.3 show the the best and average over three runs (in brackets) cost obtained by SA with and without the decomposed assignment approach. All experiments presented in this section refer to two loops performed by the decomposed assignment approach (the number of evaluations shown referring to each single loop).

Table A.7: SA - Siemens1-4 for SA without decomposition

first loop				
second loop				
noEvaluations	SIEMENS1	SIEMENS2	SIEMENS3	SIEMENS4
$10,000 * \lvert V\rvert$	4.26 (4.29)	19.36 (19.99)	10.64 (10.85)	102.00 (103.01)
	4.24 (4.27)	**18.94 (19.03)**	**10.22 (10.58)**	**100.49 (101.58)**
$100,000 * \lvert V\rvert$	3.38 (3.44)	17.12 (17.67)	8.24 (8.52)	92.43 (92.71)
	3.30 (3.37)	**16.75 (16.89)**	**8.14 (8.31)**	**92.12 (92.23)**
$1,000,000 * \lvert V\rvert$	2.84 (2.91)	15.72 (15.79)	6.80 (6.94)	90.33 (91.04)
	2.75 (2.83)	**15.72 (15.79)**	**6.61 (6.72)**	**89.25 (89.98)**
$2,000,000 * \lvert V\rvert$	2.75 (2.83)	15.72 (15.79)	6.61 (6.72)	87.25 (87.98)
	2.68 (2.76)	**15.59 (15.64)**	**6.59 (6.62)**	**86.59 (87.12)**

Table A.11, A.3, and A.3 show the the best and average over three runs (in brackets) cost obtained by SA with and without the decomposed assignment approach. For the hardest of the Siemens instances (Siemens 4) the GA results will be omitted for the shortest runs, since the algorithm does not produced any feasible solutions.

Table A.3 show the the best and average over three runs (in brackets) cost obtained by the order-based GA for Siemens1 with a decomposition into one to five subsets.

Table A.8: SA - Siemens1-4 Balanced GPFAP decomposition

noEvals.	noSub.	SIEMENS1	SIEMENS2	SIEMENS3	SIEMENS4
		first loop			
		second loop			
$10,000 * \lvert V \rvert$	2	3.91 (3.94)	19.38 (19.66)	9.18 (9.32)	99.03 (101.21)
		3.81 (3.90)	**19.23 (19.42)**	9.02 (9.17)	**98.92 (100.68)**
	3	4.28 (4.31)	21.29 (21.68)	8.92 (9.15)	102.28 (103.15)
		3.99 (4.16)	19.77 (20.04)	**8.64 (8.89)**	99.29 (100.83)
	4	4.02 (4.12)	21.94 (22.04)	9.52 (9.70)	104.95 (1,439)
		3.82 (4.01)	20.15 (20.43)	9.41 (9.49)	102.41 (770.1)
	5	4.58 (4.60)	22.38 (22.54)	10.38 (10.60)	107.97 (2,773)
		4.44 (4.49)	20.67 (21.30)	10.05 (10.45)	105.58 (106.15)
$100,000 * \lvert V \rvert$	2	3.16 (3.28)	17.91 (17.98)	7.66 (7.84)	90.65 (91.50)
		3.14 (3.18)	17.83 (17.91)	7.58 (7.81)	**90.44 (91.48)**
	3	3.36 (3.45)	19.79 (19.94)	7.30 (7.45)	93.13 (93.85)
		3.32 (3.36)	17.99 (18.09)	**7.24 (7.40)**	92.88 (93.56)
	4	3.53 (3.58)	17.72 (17.93)	8.09 (8.28)	94.72 (95.68)
		3.15 (3.19)	**17.31 (17.42)**	7.98 (8.14)	93.19 (93.38)
	5	3.90 (4.03)	17.88 (18.25)	8.75 (8.88)	96.32 (96.87)
		3.50 (3.57)	17.65 (17.79)	8.41 (8.52)	94.30 (95.19)
$1,000,000 * \lvert V \rvert$	2	2.78 (2.80)	17.04 (17.15)	6.59 (6.77)	85.24 (85.71)
		2.73 (2.77)	**16.49 (16.87)**	6.52 (6.61)	**85.18 (85.42)**
	3	2.97 (3.00)	19.65 (19.79)	6.49 (6.61)	89.96 (90.70)
		2.88 (2.94)	17.66 (17.91)	**6.42 (6.59)**	89.55 (90.27)
	4	2.99 (3.04)	20.87 (21.02)	7.03 (7.10)	90.57 (91.06)
		2.96 (2.99)	18.30 (18.38)	7.02 (7.04)	90.02 (90.71)
	5	3.48 (3.54)	20.35 (20.55)	7.68 (7.89)	93.72 (93.99)
		3.22 (3.33)	18.17 (18.34)	7.46 (7.58)	92.47 (93.18)
$2,000,000 * \lvert V \rvert$	2	2.68 (2.75)	16.97 (17.04)	6.39 (6.46)	84.35 (84.80)
		2.60 (2.69)	**16.34 (16.73)**	**6.37 (6.44)**	**84.08 (84.39)**
	3	2.95 (3.01)	19.54 (19.77)	6.53 (6.81)	89.50 (90.59)
		2.93 (2.95)	17.41 (17.60)	6.46 (6.58)	87.16 (88.51)
	4	2.94 (3.01)	20.56 (20.87)	6.95 (7.02)	89.96 (90.43)
		2.90(2.98)	18.02 (18.27)	6.79 (6.89)	89.53 (90.17)
	5	3.34 (3.43)	20.31 (20.46)	7.65 (7.79)	92.94 (93.5)
		3.04(3.13)	17.90 (18.53)	7.26 7.(41)	91.83 (92.77)

Table A.9: SA - Siemens1-4 Unbalanced GPFAP decomposition

noEvals.	noSub.	SIEMENS1	SIEMENS2	SIEMENS3	SIEMENS4
		first loop			
		second loop			
$10,000 \cdot \lvert V \rvert$	2	3.90 (4.01)	19.93 (20.17)	9.10 (9.36)	100.09 (100.66)
		3.88 (4.01)	**18.90 (19.13)**	**8.84 (9.08)**	**99.75 (100.34)**
	3	4.08 (4.16)	21.24 (21.50)	9.60 (9.81)	103.12 (104.03)
		3.96 (4.02)	19.77 (19.89)	9.24 (9.51)	101.47 (101.85)
	4	4.68 (4.79)	19.43 (19.48)	9,46 (9.84)	105.48 (1,050)
		4.14 (4.23)	18.91 (19.17)	9.54 (9.92)	103.12 (773.9)
	5	6.27 (6.38)	20.60 (20.73)	10.07 (10.28)	109.87 (2,098)
		4.29 (4.63)	19.09 (19.30)	9.65 (9.89)	106.79 (107.74)
$100,000 \cdot \lvert V \rvert$	2	3.38 (3.44)	18.24 (18.62)	7.59 (7.69)	91.91 (92.25)
		3.30 (3.37)	17.59 (17.86)	**7.31 (7.49)**	**91.14 (91.63)**
	3	3.35 (3.43)	19.79 (19.94)	7.82 (7.99)	93.39 (94.16)
		3.29 (3.34)	17.99 (18.10)	7.79 (7.98)	92.99 (93.61)
	4	3.99 (4.20)	17.72 (17.93)	8.43 (8.57)	94.84 (95.22)
		3.30(3.44)	**17.31 (17.42)**	7.92 (8.13)	93.78 (94.15)
	5	5.66 (5.85)	18.80 (19.14)	8.61 (8.86)	97.09 (97.74)
		3.91 (4.07)	17.69 (17.75)	7.69 (8.11)	95.41 (96.33)
$1,000,000 \cdot \lvert V \rvert$	2	2.74 (2.79)	17.32 (17.53)	6.31 (6.39)	86.14 (86.52)
		2.83 (2.90)	16.81 (16.90)	**6.22 (6.30)**	**85.67 (86.33)**
	3	2.74 (2.82)	19.35 (19.38)	6.93 (7.08)	91.28 (91.77)
		2.73 (2.76)	17.00 (17.16)	6.88 (6.91)	90.58 (91.01)
	4	4.02 (4.04)	16.90 (17.01)	7.28 (7.79)	91.17 (91.86)
		2.90 (3.01)	**16.15 (16.23)**	7.17 (7.21)	90.19 (90.91)
	5	5.39 (5.59)	18.69 (18.78)	7.28 (7.57)	93.25 (93.71)
		3.47 (3.53)	16.44 (16.60)	6.84 (7.02)	92.62 (92.23)
$2,000,000 \cdot \lvert V \rvert$	2	2.69 (2.74)	16.94 (17.13)	6.22 (6.23)	85.72 (85.90)
		2.61 (2.66)	16.20 (16.37)	**5.98 (6.13)**	**84.58 (85.13)**
	3	2.63 (2.75)	19.06 (19.17)	6.68 (6.77)	91.47 (91.94)
		2.63 (2.72)	16.83 (16.97)	6.42 (6.56)	89.51 (90.35)
	4	3.73 (3.85)	16.56 (16.72)	6.84 (6.88)	90.71 (91.52)
		2.85 (2.86)	**16.07 (16.27)**	6.75 (6.80)	90.02 (90.68)
	5	4.31 (4.48)	18.44 (18.57)	7.23 (7.24)	93.72 (94.04)
		3.28 (3.43)	16.30 (16.37)	6.83 (6.94)	91.18 (92.13)

Table A.10: order based GA - Siemens1 - Balanced GPFAP decomposition

noSub.	$10,000 \cdot \lvert V \rvert$ evals.	$100,000 \cdot \lvert V \rvert$ evals.	$1,000,000 \cdot \lvert V \rvert$ evals.	$2,000,000 \cdot \lvert V \rvert$ evals.
	first loop			
	second loop			
1	5.98 (6.02)	5.24 (5.32)	5.10 (5.12)	5.16 (5.03)
	5.88 (5.95)	5.03 (5.19)	5.01 (5.03)	4.97 (4.83)
2	5.57 (5.78)	5.04 (5.13)	4.62 (4.73)	4.60 (4.66)
	5.65 (5.71)	4.97 (5.03)	4.34 (4.40)	4.27 (4.31)
3	5.73 (5.82)	4.59 (4.84)	4.35 (4.39)	4.22 (4.32)
	5.46 (5.51)	**4.20 (4.52)**	4.05 (4.12)	3.93 (4.01)
4	5.52 (5.68)	4.53 (4.70)	4.29 (4.31)	4.30 (4.37)
	5.28 (5.41)	4.28 (4.55)	**3.92 (4.05)**	**3.73 (3.81)**
5	5.88 (6.00)	4.77 (4.85)	4.27 (4.41)	4.20 (4.28)
	5.56 (5.57)	4.51 (4.53)	3.99 (4.10)	3.99 (4.02)

Table A.11: GA - Siemens1-4 for SA without decomposition

† at least 1 invalid solution * no valid solutions

noEvaluations	first loop / second loop			
	SIEMENS1	SIEMENS2	SIEMENS3	SIEMENS4
$10,000 * \|V\|$	6.18 (6.40)	21.93 (22.30)	12.56 (679.40) †	–
	6.09 (6.29)	**21.81 (22.15)**	**12.45 (12.65)**	–
$100,000 * \|V\|$	5.35 (5.47)	19.73 (19.98)	12.07 (12.29)	–
	4.88 (5.00)	**19.10 (19.41)**	**11.99 (12.23)**	–
$1,000,000 * \|V\|$	4.19 (4.36)	18.20 (18.49)	10.77 (11.48)	106.95 (2,106)
	3.96 (4.01)	**18.00 (18.14)**	**9.77 (10.20)**	**103.26 (103.51)**
$2,000,000 * \|V\|$	3.96 (4.01)	18.00 (18.14)	9.77 (10.20)	105.69 (106.42)
	3.61 (3.72)	**17.64 (17.91)**	**9.54 (9.77)**	**101.05 (102.14)**

Table A.12: GA - Siemens1-4 GA Balanced GPFAP decomposition

† at least 1 invalid solution * no valid solutions

noEvals.	noSub.	SIEMENS1	SIEMENS2	SIEMENS3	SIEMENS4
		first loop / second loop			
$10,000 * \|V\|$	2	5.97 (6.05)	23.20 (23.34)	8.80 (8.84)	–
		5.91 (5.99)	**22.49 (23.12)**	8.76 (8.82)	–
	3	6.18 (6.24)	24.86 (24.95)	8.46 (8.63)	–
		5.99 (6.00)	22.97 (23.52)	**8.41 (8.53)**	–
	4	6.78 (6.84)	24.50 (24.96)	8.88 (9.07)	–
		6.05 (6.18)	23.75 (24.44)	8.76 (8.79)	–
	5	7.02 (7.17)	24.66 (24.90)	9.54 (9.79)	–
		6.63 (6.58)	22.60 (23.59)	9.11 99.31)	–
$100,000 * \|V\|$	2	4.85 (5.01)	20.82 (21.05)	7.21 (7.38)	–
		4.61 (4.72)	**19.69 (19.74)**	7.14 (7.26)	–
	3	5.03 (5.05)	22.22 (22.28)	7.28 (7.31)	–
		4.42 (4.46)	20.16 (20.24)	7.11 (7.20)	–
	4	5.18 (5.26)	22.15 (22.44)	7.40 (7.56)	–
		4.60 (4.76)	19.99 (20.18)	**6.96 (7.20)**	–
	5	6.96 (7.04)	21.92 (22.24)	8.41 (8.45)	–
		5.06 (5.32)	19.73 (19.97)	7.61 (7.75)	–
$1,000,000 * \|V\|$	2	4.14 (4.19)	19.04 (19.16)	6.39 (6.64)	106.67 (773.3)
		3.61 (3.74)	**17.94 (18.01)**	**6.20 (6.34)**	**102.09 (104.28)**
	3	3.93 (4.43)	20.88 (21.14)	6.46 (7.30)	108.92 (776.5)
		3.54 (3.63)	18.98 (19.13)	6.38 (7.63)	104.35 (704.51)
	4	4.70 (5.89)	21.39 (21.64)	6.87 (7.95)	111.63 (741.8)
		3.62 (3.78)	19.01 (19.23)	6.49 (7.55)	106.51 (738.1)
	5	8.02 (8.17)	21.35 (21.49)	7.57 (8.61)	2,075 (2,114)
		6.63 (6.58)	19.26 (19.43)	6.69 (8.00)	110.61 (687.2)
$2,000,000 * \|V\|$	2	3.73 (3.80)	18.88 (19.03)	6.11 (6.65)	98.31 (99.93)
		3.40 (3.45)	**17.83 (17.86)**	**6.08 (6.21)**	**96.84 (97.22)**
	3	3.7 (3.75)	20.70 (20.95)	6.33 (7.09)	103.61 (104.40)
		3.30 (3.44)	18.70 (18.89)	6.21 (7.05)	101.09 (101.93)
	4	4.63 (4.81)	21.50 (21.79)	6.74 (7.34)	102.55 (103.60)
		3.37(3.49)	19.10 (19.21)	6.28 (7.19)	101.84 (102.19)
	5	5.70 (6.28)	21.07 (21.23)	7.32 (8.06)	110.71 (768.1) †
		4.45 (4.61)	18.95 (18.97)	6.66 (7.14)	106.81 (710.8) †

Table A.13: GA - Siemens1-4 GA Unbalanced GPFAP decomposition

† at least 1 invalid solution * no valid solutions

noEvals.	noSub.	SIEMENS1	SIEMENS2	SIEMENS3	SIEMENS4
		first loop			
		second loop			
10,000 • $\|V\|$	2	6.07 (6.12)	23.02 (23.28)	8.60 (8.75)	–
		5.86 (5.97)	**22.55 (23.07)**	**8.42 (8.53)**	–
	3	6.20 (6.31)	24.42 (24.84)	8.88 (9.14)	–
		6.15 (6.23)	23.46 (24.07)	8.75 (8.93)	–
	4	5.97 (6.07)	22.87 (23.27)	9.21 (9.52)	–
		5.82 (5.93)	22.52 (23.08)	8.89 (8.96)	–
	5	6.42 (6.58)	24.27 (24.29)	9.68 (9.75)	–
		6.08 (6.27)	22.51 (23.30)	9.07 (9.17)	–
100,000 • $\|V\|$	2	4.69 (4.88)	21.15 (21.72)	6.97 (7.06)	–
		4.34 (4.41)	**19.79 (20.30)**	**6.94 (7.04)**	–
	3	4.73 (4.87)	22.29 (22.41)	7.27 (7.56)	–
		3.93 (4.14)	19.95 (20.11)	7.20(7.37)	–
	4	4.45 (4.50)	20.50 (20.91)	7.65 (7.83)	–
		3.98 (3.92)	19.48 (19.81)	7.40 (7.55)	–
	5	4.84 (4.92)	21.82 (22.23)	8.10 (8.26)	–
		4.00 (4.08)	19.98 (20.11)	7.44 (7.59)	–
1,000,000 • $\|V\|$	2	3.72 (3.79)	18.69 (19.27)	6.82 (6.91)	105.54 (106.03)
		3.39 (3.46)	**17.61 (18.05)**	**6.63 (6.75)**	104.13 (104.68)
	3	3.59 (3.70)	20.56 (20.98)	7.13 (7.67)	104.08 (104.18)
		2.98 (3.20)	18.19 (18.68)	6.78 (6.92)	**103.69 (103.95)**
	4	3.57 (3.62)	18.96 (19.09)	7.50 (7.99)	104.41 (105.09)
		3.15(3.21)	17.93 (18.28)	6.81 (7.06)	103.83 (103.99)
	5	3.90 (3.98)	20.44 (20.49)	8.01 (8.62)	104.70 (105.12)
		3.14 (3.30)	18.91 (18.95)	7.41 (7.74)	104.17 (104.66)
2,000,000 • $\|V\|$	2	3.53 (3.57)	18.96 (19.38)	6.34 (6.62)	104.49 (104.88)
		3.18 (3.28)	**17.46 (18.21)**	**6.17 (6.41)**	103.64 (103.68)
	3	3.31 (4.52)	20.57 (20.85)	7.02 (7.49)	103.11 (103.53)
		3.07 (3.14)	18.07 (18.44)	6.69 (6.77)	**102.89 (103.02)**
	4	3.46 (3.49)	18.87 (19.11)	7.37 (7.83)	103.52 (104.15)
		2.86 (3.04)	17.85 (18.23)	6.76 (6.88)	102.90 (103.24)
	5	3.75 (3.83)	20.03 (21.55)	7.43 (7.52)	104.71 (105.28)
		3.24 (3.27)	18.42 (18.79)	7.10 (7.06)	104.53 (104.84)

Table A.15, A.14, and A.16 show the best and average over three runs cost for the Bradford benchmarks obtained by SA with and without the decomposed assignment approach. For all of the instances we present the results produced by the balanced GPFAP after $100,000 * |V|$ and $2,000,000 * |V|$ total evaluations. For Bradford0, Bradford2, and Bradford4 we also present for comparison the results produced by the unbalanced version.

Table A.17 summarizes the results for the Siemens1 and Bradford0 experiments performed by SA with a balanced GPFAP decomposition in which the partitioning considers only the soft constraints, see Section 7.3.2. We present the best and average (in brackets) cost over three runs obtained after $100,000 * |V|$ and $2,000,000 * |V|$ total evaluations for a decomposition into two to five subsets. For the results with the whole approach we refer for comparison to Tables A.7 and A.15.

Table A.14: SA - Bradford - GPFAP decomposition

noEvals.	noSub.	BRADFORD0		BRADFORD2	
		Bal. GPFAP	Unbal. GPFAP	Bal. GPFAP	Unbal. GPFAP
		first loop			
		second loop			
$100,000 * \|V\|$	2	2.04 (2.25)	2.40 (2.44)	7.15 (7.73)	6.72 (6.93)
		1.92 (1.99)	1.97 (2.08)	6.27 (6.40)	**5.41 (5.58)**
	3	1.83 (1.87)	1.99 (2.01)	7.63 (8.24)	7.49 (8.17)
		1.67 (1.80)	1.92 (1.94)	7.32 (7.41)	6.89 (7.26)
	4	2.23 (2.28)	2.48 (2.83)	8.26 (8.82)	8.21 (8.79)
		2.13 (2.17)	2.00 (2.09)	7.65 (7.84)	7.62 (7.77)
	5	2.90 (2.94)	3.38 (3.39)	8.17 (8.91)	8.07 (8.64)
		2.41 (2.43)	1.91 (2.04)	8.27 (8.35)	7.37 (7.52)
$2,000,000 * \|V\|$	2	1.31 (1.41)	1.37 (1.72)	5.17 (5.36)	4.79 (4.99)
		1.16 (1.23)	1.09 (1.14)	4.27 (4.24)	**3.83 (3.92)**
	3	1.18 (1.20)	1.11 (1.18)	6.45 (6.92)	6.15 (6.21)
		1.07 (1.13)	1.09 (1.14)	5.34 (5.49)	5.05 (5.24)
	4	1.49 (1.63)	2.17 (2.31)	6.97 (7.17)	6.25 (6.34)
		1.26 (1.37)	1.24 (1.26)	5.75 (5.93)	5.14 (5.33)
	5	2.07 (2.15)	2.62 (2.72)	7.25 (7.64)	6.79 (7.24)
		1.47 (1.54)	1.26 (1.29)	6.13 (6.58)	5.96 (6.57)

Table A.15: SA -Bradford0-10 for SA without decomposition

	first loop				
	second loop				
noEvaluations	BRADFORD0	BRADFORD1	BRADFORD2	BRADFORD4	BRADFORD10
$100,000 * \|V\|$	2.40 (2.46)	2.83 (2.87)	6.69 (6.85)	27.39 (27.72)	196.53 (197.05)
	2.16 (2.37)	**2.32 (2.41)**	**6.57 (6.71)**	**27.19 (27.24)**	**195.48 (196.36)**
$2,000,000 * \|V\|$	1.42 (1.66)	1.96 (2.03)	4.56 (4.71)	20.84 (21.07)	188.64 (190.08)
	1.31 (1.43)	**1.64 (1.75)**	**4.46 (4.53)**	**20.62 (20.79)**	**187.13 (188.34)**

Table A.16: SA - Bradford - GPFAP decomposition

noEvals.	noSub.	BRADFORD4		BRADFORD1	BRADFORD10
		Bal. GPFAP	Unbal. GPFAP	Bal. GPFAP	Bal. GPFAP
		first loop			
		second loop			
$100,000 * \|V\|$	2	27.53 (27.68)	27.72 (27.93)	3.21 (3.24)	201.26 (202.56)
		25.19 (25.39)	26.45 (26.48)	**1.88 (1.91)**	**192.21 (193.56)**
	3	31.09 (31.62)	31.37 (31.79)	4.10 (4.42)	223.90 (225.92)
		29.43 (29.86)	30.00 (30.21)	2.55 (2.59)	206.50 (208.38)
	4	32.89 (33.68)	33.95 (34.12)	5.82 (5.92)	234.41 (235.16)
		31.37 (32.08)	32.38 (32.89)	3.16 (3.20)	214.90 (216.53)
	5	35.19 (36.45)	35.83 (36.18)	6.06 (6.57)	235.81 (236.09)
		26.01 (26.52)	27.36 (28.19)	4.25 (4.62)	228.26 (230.04)
$2,000,000 * \|V\|$	2	21.40 (21.83)	22.17 (22.77)	2.16 (2.39)	192.37 (192.98)
		18.51 (18.78)	19.95 (20.44)	**1.32 (1.21)**	**185.05 (185.39)**
	3	25.83 (26.27)	26.64 (27.81)	2.46 (2.80)	203.69 (204.60)
		20.54 (21.01)	21.66 (22.21)	1.88 (2.11)	188.38 (189.51)
	4	33.93 (34.29)	34.82 (35.29)	3.38 (3.65)	211.81 (212.94)
		23.09 (24.65)	24.63 (25.50)	2.36 (2.56)	193.51 (194.43)
	5	37.85 (38.14)	38.71 (39.31)	4.34 (4.97)	213.90 (215.56)
		28.52 (29.83)	29.68 (30.56)	3.68 (3.81)	196.02 (197.23)

Table A.17: Siemens1 and Bradford0 - Balanced GPFAP decomposition with only soft constraints

noSub.	$10,000 * \|V\|$ evals.	$100,000 * \|V\|$ evals.	$1,000,000 * \|V\|$ evals.	$2,000,000 * \|V\|$ evals.
	first loop			
	second loop			
SIEMENS1				
1	4.26 (4.29)	3.38 (3.44)	2.84 (2.91)	2.75 (2.83)
	4.24 (4.27)	3.30 (3.37)	2.75 (2.83)	2.68 (2.76)
2	3.92 (4.11)	3.21 (3.34)	2.83 (2.88)	2.72 (2.91)
	3.75 (3.89)	3.14 (3.29)	2.76 (2.81)	**2.60 (2.69)**
3	3.94 (4.17)	3.15 (3.21)	2.82 (2.89)	2.77 (2.88)
	3.77 (3.90)	**3.12 (3.20)**	**2.70 (2.77)**	2.69 (2.75)
4	4.03 (4.15)	3.25 (3.32)	2.92 (3.00)	2.76 (2.88)
	4.01 (4.07)	3.20 (3.28)	2.67 (2.87)	2.68 (2.79)
5	4.04 (4.24)	3.35 (3.39)	2.88 (2.94)	2.87 (2.94)
	3.93 (4.02)	3.24 (3.31)	2.78 (2.86)	2.73 (2.85)
BRADFORD0				
1	3.69 (3.82)	2.40 (2.46)	1.53 (1.88)	1.42 (1.66)
	3.60 (3.75)	2.16 (2.37)	1.42 (1.66)	1.31 (1.43)
2	3.19 (3.29)	1.92 (1.95)	1.28 (1.32)	1.09 (1.14)
	3.02 (3.17)	1.89 (1.91)	1.20 (1.23)	**1.04 (1.10)**
3	3.15 (3.38)	1.81 (1.91)	1.11 (1.22)	1.17 (1.21)
	3.01 (3.16)	**1.79 (1.90)**	**1.10 (1.16)**	1.08 (1.10)
4	4.20 (4.37)	2.98 (3.06)	2.36 (2.52)	2.22 (2.30)
	3.21 (3.30)	2.08 (2.15)	1.54 (1.62)	1.44 (1.47)
5	3.78 (4.00)	2.86 (2.95)	2.18 (2.24)	1.77 (1.93)
	3.35 (3.52)	2.36 (2.43)	1.71 (1.73)	1.59 (1.68)

Appendix B

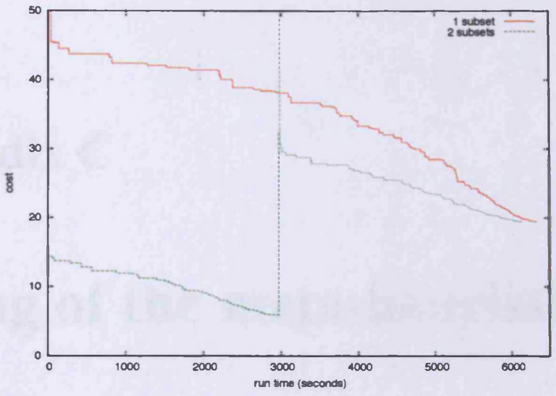
Condor Pool

B.1 The Cardiff University Condor Pool

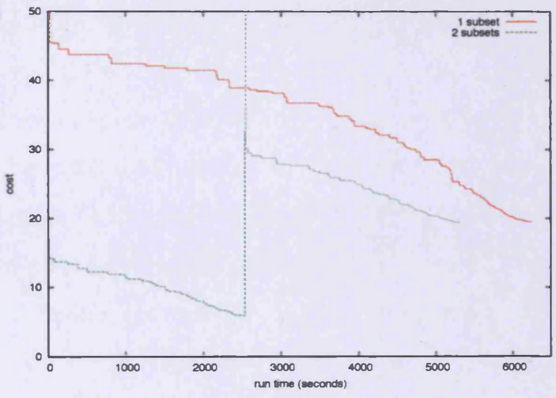
The Condor Pool is the product of the Condor Research Project at the University of Wisconsin-Madison, and it was first installed as a production system in the UW-Madison Department of Computer Sciences in the early nineties. This project aims to develop, implement, deploy, and evaluate mechanisms and policies that support High Throughput Computing (HTC) on large collections of distributively owned computing resources. Software tools have been developed to enable scientists and engineers to increase their computing throughput. Users submit their serial or parallel jobs to Condor, which places them into a queue, chooses when and where to run the jobs based upon a policy, carefully monitors their progress, and ultimately informs the user upon completion.

Cardiff University has a large scalable Windows Condor pool, which distributes Condor to 'Open Access' workstations to provide a centrally managed High Throughput Computing service to all researchers of the University. There are currently 800 workstations owned by Information Services throughout the whole University campus. They are all Windows XP SP1 and access the main Novell NetWare network.

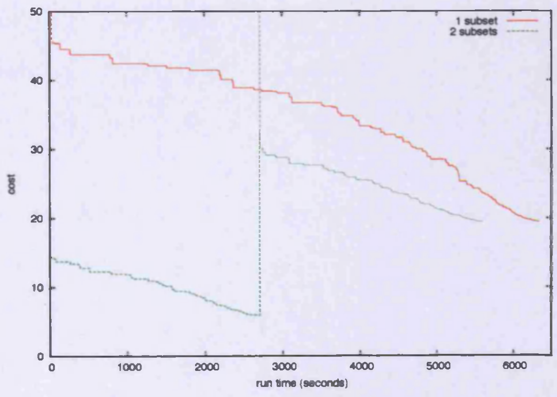
We have implemented a procedure, which runs for all of the experiments conducted on a specific benchmark, in order to compare the cost-run time output produced by different machines in the pool. The procedure performs 100 fitness evaluations of a sample assignment generated at random and records the computational time in millisecond. The same run-time recording is also executed (for a fixed values of the random seeds) for both the whole and the decomposed assignment approach on a specific test machine (a 3.000 GHz Intel Pentium 4). Subsequently, all run times produced by the different machines in the Condor pool are scaled with the reference value obtained in that test. Figure B.1 shows the cost-runtime plot for a run of Siemens 2 (with the whole and the decomposed assignment approach into two subsets) produced respectively by the Condor pool, the test machine, and the final scaled output obtained by the procedure described above.



Condor pool



test machine



scaled output

Figure B.1: Cost-time plot for Siemens2 solved by SA with the whole approach and GP-FAP decomposition into two subsets ($1,000,000 * |V|$ evaluations)

Appendix C

Tuning of the meta-heuristics

C.1 Tuning for FS-FAP

The advantage of the permutation based GA in Algorithm 5.1 applied to the MS-FAP was the property of generating only zero-violations assignments, thus reducing significantly the search space and speeding up its convergence without preventing the full exploration of the solution space. For the FS-FAP, the domain of available frequencies is limited and it is usually not possible to find a zero-violation assignment for a given set of transmitters. As a consequence, the evaluation of an individual is computed according to the modified sequential algorithm proposed in Algorithm 6.1. However, some other preliminary tests were needed in order to determine an appropriate setting of the parameters.

It is known that one of the most difficult parameter to set in a GA is the size of the population and the consequent number of generations required to converge to a near optimal solution. This choice depends on the individual problem considered and affects the results either in terms of solution quality or runtime. We can sensibly expect a relationship between the optimal population size and the size of the problem considered, namely the number of transmitters. A number of experiments were conducted using the test problems P_1, P_2 and C_4. The number of frequencies available for each of the instances was chosen relative to the minimum span values presented in Table 5.1. Initial experiments varied the population size, using the number of transmitters $|V|$, together with $2 \times |V|$ and $\frac{|V|}{2}$. For the smaller problems greater population sizes have been tested too. The GA was stopped when there were no further improvements for 100 generations. Results are summarized in Table C.1, which gives the mean and the best costs after three runs for each of the test problems considered. Note that, because of the tendency of the GA to get trapped in local minima (see the phenomenon of the genetic drift [48]), increasing the number of generations for a fixed population size does not often produce any improvement in the quality of solutions. Although the tests performed indicate that best performance are produced by the biggest population, considerations about the percentage improvement and the run times required has led to the choice of keep-

		P_1 20 freq.		P_2 300 freq.		C_4 50 freq.	
Pop. Size	Gen.	Mean	Best	Mean	Best	Mean	Best
500	500	132.7	132	223.7	217	303.3	292
1000	500	132	**130**	208.3	205	299.3	297
1000	1000	131.3	**130**	206.7	205	298.3	296
2000	2000	**130**	**130**	**205.7**	**203**	**295.7**	**293**

Table C.1: Mean and best cost for data sets P_1, P_2 and C_4 with different population sizes

Mut. Cross.	one per offspring	0.01	0.005
100%	132	132	**131**
	130	**130**	**131**
80%	134	134.7	134.7
	132	134	134
50%	134.3	132.3	135
	133	131	135

Table C.2: Best cost for problem P_1 with a fixed spectrum of 20 and different Crossover and Mutation rates.

ing the same settings already used for the MS-FAP in Section 5.1 (500 generations and a size of 1000 chromosomes).

In [118] the genetic operators are applied with rates that produce the best performance for the MS-FAP. Cycle crossover was applied with a rate of 100% whereas one single order-based mutation was performed for each of the offspring generated. Different rates have been tried on data set P_1 for the FS-FAP, increasing respectively the amount of mutation and decreasing the crossover. In particular, crossover was applied with rates of 50%, 80%, and 100% while mutation has been tested with rates of 0.01 and 0.005. These values were chosen according to previous works on GAs with order-based representation (see [23]). Results are presented in Table C.2. No significant improvements were observed and so the original rates were kept for all of the rest of the experiments performed.

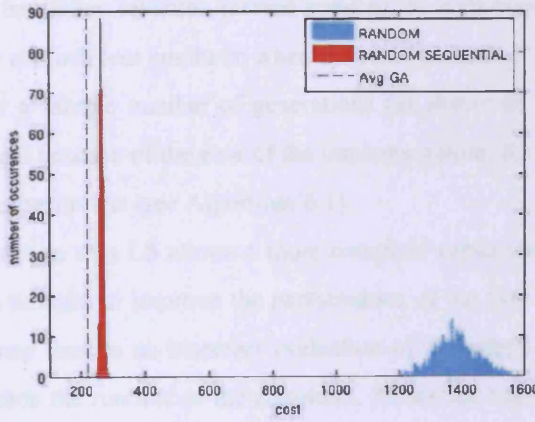
To evaluate the effectiveness of the GA we compared its results with a series of runs of random assignments and sequential assignments applied to random permutations. As an example, Table C.3 compares for test problem P_2 the mean cost

	GA	Random	Random + Sequential
P_1 20 freq.	**132.0**	395.0	175.7
P_1 40 freq.	**14**	202.7	28.7
P_2 300 freq.	**208.3**	1378.7	263.3
P_2 400 freq.	**27.0**	1034.0	57.3
C_1 50 freq.	**349.7**	2317.0	594.0
C_1 65 freq.	**96.2**	1786.3	280.3
C_3 50 freq.	**299.3**	2313.3	528.3
C_3 65 freq.	**47.3**	1784.3	221.7

Table C.3: Comparison with Random and Random Sequential assignments

over five runs produced by the GA with the mean values produced by a thousand random assignments, and by the same number of random orderings followed by the sequential algorithm 6.1. For the same problem, Figure C.1 plots the graph cost versus number of occurrences obtained by the random assignments above and compares them with the mean costs produced by the GA.

Figure C.1: Cost versus number of occurrences for data set P_2 with a fixed spectrum of 300 channels



Further experiments have been conducted to determine which of the initial orderings originally proposed in [54] produces the lower cost using the sequential assignment. The same ordering will be used in order to generate the initial population of the order-based GA and as the basic ordering to be further decomposed by the generalized degree decomposition using the classification in [110]. We consider *Largest Degree First* (LF), *Largest Degree First (Exclusive)* (LFE), *Smallest Degree Last* (SL) together with their generalized versions (GLF, GLFE, GSL). During the frequency allocation transmitters will be selected only sequentially from the given initial ordering.

In addition, the effectiveness of the algorithm which selects the next frequency to be assigned can also be investigated. Note that, the sequential algorithm 6.1 only modifies the *Smallest Acceptable Frequency* (SAF) procedure, by restricting it to the the choice of the channel which produces the lowest constraint violations. Alternatively, other algorithms can be used, such as *Smallest Acceptable Occupied* (SAO, which further restricts the choice to the already occupied frequencies), and *Smallest Most Heavily Occupied* (SAMHO, which assigns the smallest among the most heavily occupied channels). Table C.4 shows the results for three benchmarks, in which it is shown the number of violations *violNo* together with the cost already defined in Problem 1.2 (that is the sum of total violations of the interfering transmitters). From the figures it appears that (as for the MS-FAP) GSL is the best performing algorithm for the selection of the next transmitter selection. SAMHO is superior for frequency selection instead some of the tests conducted. However, this superiority is much less emphatic when the same procedure is used to actually run the GA for a sample number of generations (as shown in Table C.5). As a consequence, and because of the ease of the implementation, SAF was used for all the rest of the experiments (see Algorithm 6.1).

The introduction of a LS allows a more complete exploration of the solution space and it is thought to improve the performance of the GA. However, too intensive a LS may lead to an incorrect evaluation of the effectiveness of the GA itself, and degrade the runtime of the algorithm. Hence the basic LS procedure in Algorithm 6.2 has been implemented with the simple acceptance criteria proposed in Definition 6.3 (*hill climbing* and *Metropolis*).

Ordering	Assignment	$\sum_{e\in E(G)} \varphi_{FS}(f,e)$	violNo	$\sum_{e\in E(G)} \varphi_{FS}(f,e)$	violNo	$\sum_{e\in E(G)} \varphi_{FS}(f,e)$	violNo
		P_1 35 freq.		P_2 400 freq.		C_1 65 freq.	
LF	SAF	44	24	58	58	158	153
LF	SAO	49	29	139	84	176	167
LF	SAMHO	49	26	134	134	168	160
LFE	SAF	45	29	63	63	173	159
LFE	SAO	46	18	141	85	165	154
LFE	SAMHO	46	22	135	74	278	239
SL	SAF	42	19	58	58	289	236
SL	SAO	46	19	141	85	277	226
SL	SAMHO	42	19	134	134	155	153
GLF	SAF	48	22	63	63	155	150
GLF	SAO	50	19	68	34	157	151
GLF	SAMHO	49	21	68	34	169	159
GLFE	SAF	46	25	68	34	254	192
GLFE	SAO	46	25	63	63	249	179
GLFE	SAMHO	46	27	63	63	264	205
GSL	SAF	**38**	**17**	**44**	34	156	154
GSL	SAO	40	20	47	32	157	151
GSL	SAMHO	39	19	**44**	**32**	**133**	**133**

Table C.4: Comparison of sequential assignments algorithms for benchmarks P_1, P_2, and C_1

Ordering	Assignment	$\sum_{e\in E(G)} \varphi_{FS}(f,e)$		
		P_1 35 freq.	P_2 400 freq.	C_1 65 freq.
GSL	SAF	**29**	**27**	100
GSL	SAO	30	**27**	131
GSL	SAMHO	**29**	**27**	**98**

Table C.5: Comparison of different sequential assignment algorithms for the order-based GA run for 500 generations. Results for data sets P_1, P_2, and C_1

Firstly, the LS has been tested for a number of times on some good assignments, such as those produced by the original GA after few generations, without being able to produce any cost improvement. This confirms the actual effectiveness of the GA procedure itself. Then the LS has been included into the GA after the individuals fitness evaluation, thus applied to each of the chromosome in the initial population and to each of the offspring newly generated by the genetic operators during the run of the algorithm. HC and MET have been initially tested on the data sets P_1 with fixed spectrums of 20 and 40 channels, and P_2 with a fixed spectrum of 300 and 400 frequencies. A value of k equal to 1 and different values of T in the range 1-10 have been used for the Metropolis algorithm. To better analyze the effectiveness of the LS we let the algorithm run for longer (i.e 1000 generations). Table C.6 summarizes the results and compares them with those given by the original GA procedure without any LS applied.

	P_1 20 freq.		P_1 40 freq.		P_2 300 freq.		P_2 400 freq.	
LS	Mean	Best	Mean	Best	Mean	Best	Mean	Best
MET T=10	**130**	**129**	**13.7**	**13**	**207.0**	204	**27**	**27**
MET T=1	136	135	14	14	210.7	210	**27**	**27**
MET T=0.5	135.7	135	14	**13**	207.3	204	**27**	**27**
MET T=0.3	132	**129**	15	15	**207**	203	**27**	**27**
MET T=0.1	131	131	14.3	14	209	203	**27**	**27**
HC	131	**129**	14	14	206.3	**201**	**27**	**27**
No LS	132	130	14.7	14	208.3	205	**27**	**27**

Table C.6: Mean and best cost for data sets P_1, and P_2 with different LS

A first important consideration is that the LS is not always effective and sometimes produces a worse performance than the original GA. It is important to stress that this happens even if the LS itself cannot actually introduce worse individuals than those in the assignments produced by the plain sequential algorithm 6.1 without any further local search procedures added. This can be explained by the following considerations. We already mentioned that the aim of the LS was to:

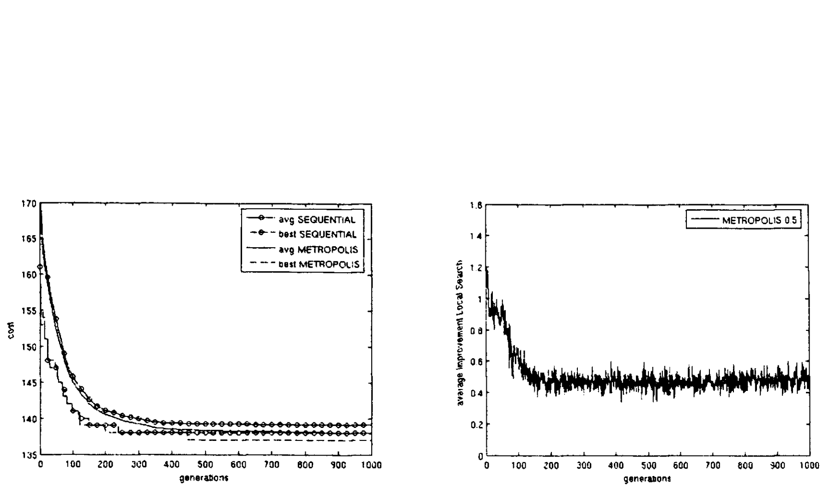
- Allow a wider exploration of the search space. Thus good assignments, already produced by the sequential algorithm, can be further improved by exploring neighbours in the solution space.

- Add more variety in the population. Thus some orderings which would have been excluded with an evaluation on a sequential basis, can be introduced into the population. This increases its variety and makes the LS actually active in the GA procedure.

If only the former statement is true, the LS does not play an active role and may result in a worse performance than the original GA procedure.

Figure C.2 shows the best and the mean cost over the population for each generation of a single run of data set P_1 with a fixed spectrum of 20 channels. LS Metropolis is applied with a value of T equal to 0.5. The figure compares these costs with those obtained by the GA without any LS added. It also shows the average cost improvement due to the LS over the generations. We can see how the LS gives a relevant contribution during the first generations only. In fact, the final cost produced is worse than the one obtained by the original GA with no search applied (see Table C.6).

Figure C.3 shows a situation when the LS is successful instead. We can see that (between 500 and 900 generations approximatively), the mean sequential cost becomes slightly worse. This means that the LS introduces new different orderings, which would have been excluded with an evaluation only based on the sequential assignment. We can also see how the mean cost improvement presents, at the same time, a discontinuity in the corresponding graph during the same range of generations. We can interpret this as a sign of the effectiveness of the LS, which results in an actual improvement in the final cost. However, Table C.6 shows how it is rather difficult to set the value of T which gives the best performance for the Metropolis algorithm. As a consequence, for the remaining experiments we chose to use a variable value of T decreasing from a high value to a low value, set respectively to 10 and 0 in our tests. Finally, Table C.7 compares the performance of the GA respectively with the final LS chosen and without it, showing the significant benefit brought about by the local improvement procedure.



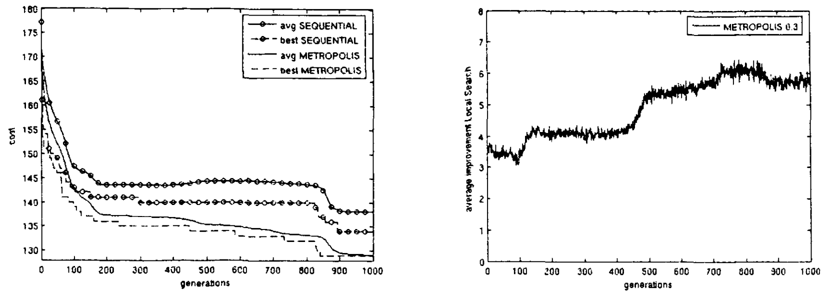
mean cost   mean cost improvement

Figure C.2: Costs produced by a single run of data set P_1 with a fixed spectrum of 20 channels and Local Search Metropolis with T=0.5

	GA - MET T 10 $\rightarrow$ 0		GA - No Search	
	Mean	Best	Mean	Best
P_1 20 freq.	**130**	**129**	132.7	130
P_1 40 freq.	**13.6**	**13**	14	14
P_2 300 freq.	**207.0**	**204**	208.3	205
P_2 400 freq.	**27**	**27**	**27**	**27**
C_1 50 freq.	**346.0**	**338**	349.7	344
C_1 65 freq.	**95.7**	**94**	96.0	96
C_3 50 freq.	**298.7**	**296**	299.3	297
C_3 65 freq.	**46.4**	**44**	47.7	45

Table C.7: GA comparison with and without LS

For some of the fixed spectrum instances (e.g. G_1, G_2 and G_3), the order based GA is outperformed by other standard meta-heuristics using the whole approach. This is essentially due to the intrinsic limitations of the order-based representation and the impossibility of increasing the amount of the local search procedure without having the natural structure of the completely distorted. Table C.7 shows a comparison of the non-decomposed approaches for SA and the GA. Consequently, since the considerable gap between the two algorithms, it becomes necessary to change the representation of the GA to the direct representation described in Section 7.1.2.



mean cost mean cost improvement

Figure C.3: Costs produced by a single run of data set P_1 with a fixed spectrum of 20 channels and Local Search Metropolis with T=0.3

C.2 Tuning for MI-FAP

C.2.1 SA

As reported in [60,100] one of the problems with SA is that its performance can be heavily affected by the choice of initial temperature and other parameters regulating the cooling phase. A possible approach to this choice is to adopt either rather high values or set the initial values according to some acceptance criteria, for example setting the temperature for which a minimum number of moves is accepted [62]. However, this value is still strictly related to the order of magnitude of the cost produced for a specific problem.

According to the terminology introduced in [62], the different elements of the SA structure used in our implementation are:

System configuration The configurations representing a solution $F = \{f_1, f_2, ..., f_n\}$ to the minimization problem representing the FAP are integer vectors in which f_i represents the frequency assigned to transmitter i, with n the total number of transmitters in the network.

Neighbour moves We considered all the moves proposed in [60] and then chose that producing the best performance for each type of FAP. In all the definitions below the new channel values are assigned to randomly chosen transmitters. Frequencies are also chosen at random within the corresponding transmitter domains.

single move Neighbours are the assignments that differ only for a single frequency value assigned to a specific component in the vector:

$$F' \rightarrow F \Leftrightarrow \exists j,\ 1 \leq j \leq n : f_j' \neq f_j \mathit{and}\ \ f_k' = f_k\ \forall\ 1 \leq k \leq n,\ k \neq j$$

double move Neighbours are the assignments that differ for exactly two frequency values assigned to two distinct components in the vector:

$$F' \rightarrow F \Leftrightarrow \exists j_1, j_2\ 1 \leq j_1, j_2, \leq n : f_{j_1}' \neq f_{j_1}, f_{j_2}' \neq f_{j_2}$$

$$\mathit{and}\ f_k' = f_k\ \forall\ 1 \leq k \leq n,\ k \neq j_1, j_2$$

restricted move Neighbours are the assignments produced by the single and double moves above in which valid moves are those producing at least one violation in the definition in eq.(1.2).

$$F' \rightarrow F \Leftrightarrow \exists i, j : |f_i - f_j| \leq c_{ij}$$

Note that for the MI-FAP the condition is equivalent to assignments that violate the hard constraints only.

Transition probability Changes from an old to a new configuration are accepted with probability $prob = e^{-\frac{C_{new} - C_{old}}{B \cdot t}}$. This criterion was applied for the first time to numerical optimization in [84]. It is known as the Boltzman distribution and simulates a thermodynamical system, in which configurations represent energy states. The constant B is also called the Boltzman constant.

Cost values The cost fitness functions $C(F)$ for a given configuration (assignment F) are set as the objectives $O_{MS,FS,MI}$ defined in Section 1.3.

Table C.8: Parameter setting for the COST259 Siemens2 benchmark

t_0	cost O_{MI}
1	20.66
0.5	20.22
0.05	17.89
0.005	16.52
0.005	18.19
0	22.5

$t_0 = 0.05$	
single move	16.52
double move	20.38
restricted move	16.73

Annealing schedule We limited the choices to the simple cooling scheme described below, in which the initial temperature t_0, the final temperature t_{min}, and the parameter α are set after a number of test runs whereas the number of iterations per loop *numL* is usually set as the number of transmitters included in the subset currently considered. If we then call *numT* the resulting number of iterations for the temperature cooling we have for each run of SA a total number of moves $numT * numL$.

For our implementation, see Algorithm 7.1, we ran a number of tests on the Siemens problems in order to set the initial temperature. Furthermore, these preliminary experiments also investigated, for the best performing temperature, the effectiveness of adopting the different types of moves described in Section 7.1.1. From these tests an initial temperature of 0.5 and the basic 'single move' produced the best outcomes, thus this setting was chosen for the rest of the experiments. As an example, Table C.8 shows the average values over three runs obtained for the Siemens2 benchmarks.

Another influential aspect of SA is the choice of the parameter regulating the cooling scheme. Note that, when decomposition is applied, in order to respect the condition in (4.1.1) we have to perform the algorithm for the same number of evaluations (that is the total number of configurations explored during the search) when we compare the solution produced by solving the original problem as a whole and by decomposition. This can be carried out in different ways.

In our approach we use a number of iterations *numLoop* equals to the number of transmitters in the current subset. Then we calculate the reduction index α, obtained using both the values of the initial (t_0) and final temperature of the annealing (defined as t_{min}), in order to satisfy the total number of evaluations required expressed by a multiple of the total number of transmitters $|V|$. This guarantees that the solutions obtained with both the whole and the decomposed assignment approach consider the same number of total evaluations, which constitutes a fair basis for comparison.

C.2.2 GA

The GA used for the MI-FAP adopts the simpler straightforward direct representation (see section 7.1.2). This representation provides ease of implementation and permits complete exploration of the search space for all of the types of FAP studied here. However, when used to solve the MS-FAP it can generate solutions which have violations and, as a consequence, the order-based representation presented in Section 5.1 is preferable for this particular problem.

With the direct representation the main difficulty arises in the choice of effective genetic operators. In fact, none of the standard operators proposed in the literature for this representation, such as one-two points crossover, uniform crossover and uniform mutation [23, 32], produces a good performance. Results can be improved by using some of the problem specific operators referenced in Section 2.2.3. In particular, best outcomes are obtained with a slightly modified version of those proposed in [68]. However, some problems still remain in the application of the crossover, since it can produce very disruptive effects and appears in general unable to preserve and transmit good characteristics to new generations. On the contrary, less disruptive operators have little effect on solutions after a few generations, when individuals start becoming very similar. As a consequence the other genetic operator, i.e. mutation, needs to assume the role of a proper local search which acts as a repair mechanism and leads the solutions towards local optima. In our implementation we have used some of the operators proposed in the literature as moves of a hill climbing procedure. However, this can increase the tendency for the whole population of getting trapped in local optima, since it becomes more difficult for the GA to move from a local minimum region to another. Finally, the suggestion of running only mutation-based GAs, see [55, 68], can prevent the genetic operators from being too disruptive, but does not seem able introduce enough variety in the population in order to avoid premature convergence.

As explained in Section 7.1.2 we apply the NGSA-II framework described in Figure 7.3. Some preliminary tests have been performed in order to find the most suitable among the limited range of crossovers and mutations available. Tables C.10 and C.9 show an example of these tests for the Siemens2 benchmark. Experiments used a sample population of 20 individuals run for a high number of generations (5000), which is one of the most commonly used sizes, since the presence of a sorting procedure can become too expensive for large populations [34].

We compared the standard uniform crossover [32], and the original version of the crossover proposed by Kapsalis et al. in [68], in which a number of non interfering edges are selected swapping the frequencies between the corresponding endpoints (our modified version is described in Algorithm 7.2). All of the crossovers were applied at a rate of 80%. Similarly, for the mutation operator we tested two different operators also proposed in [68]: *swap mutation*, see Section 7.1.2, and a more specialized one (to whom we will refer again as *kapsalis*), which consists of choosing an edge whose assignment violates a hard constraint and then changing at random the frequencies of its endpoints with a non-violating pair. It is important to note that, since the crossover still produces rather disruptive results, the mutation has also been tested as a *hill climbing* procedure (HC) [62], which actually assumes the characteristic of a *repair mechanism* and leads the solutions towards local optima (we will refer to this as mutation *swap HC*). In addition a few runs were conducted with only the mutation operator applied, as suggested in a number of works (see Section 7.1.2). From the results obtained we can draw the

Table C.9: Influence of the crossover (top) and mutation operator (bottom) for the COST259 Siemens2 benchmark

Crossover	Mutation (mut. rate)	Mean cost O_{MI}
uniform	swap (1 per individual)	12,020
uniform	swap (0.005)	950,000
uniform	swap HC (0.1)	2,030
uniform	swap HC (1.0)	24.01

Crossover	Mutation (mut. rate)	cost O_{MI}
uniform	swap HC (1.0)	24.01
kapsalis	swap HC (1.0)	23.77
kapsalis modified	swap HC (1.0)	21.74

following considerations:

- The conventional use of the mutation operators is not able to correct the disruptive effect of the crossover whereas the results improve when it is incorporated in a HC procedure. However the mutation rate, expressed as a ratio between the number of mutations and the number of transmitters, needs to be considerably high before starting to produce feasible solutions.

- Among the crossovers our modified version performs slightly better than the others tested.

- We apply swap mutation since the variations in the results are not significant while it has a simpler implementation.

- The results obtained without crossover are generally poor and only improve if a proper local search procedure is used as mutation operator (such as the 1-opt described in Algorithm 7.4) or a heavy amount of HC, for example applied at a rate of 1.0).

Since the results produced are still not fully satisfactory, we chose to restore the normal mutation operator to preserve diversity in the population (thus without the function of a local search), and add instead a proper LS procedure incorporated

Table C.10: Influence of the mutation operator for the COST259 Siemens2 benchmark

Crossover	Mutation (mut. rate)	cost O_{MI}
kapsalis modified	swap HC (1.0)	21.74
kapsalis modified	kapsalis HC (1.0)	21.56

into the GA structure, as for the order based GA. Note that the purpose behind the addition of a LS is completely different in the two cases. For the order-based representation this was necessary in order to permit full exploration of the search space, whereas for the discrete representation the complete exploration is guaranteed, and the LS has only the function of improving the GA performance. We then used the genetic operators summarized in Section 7.1.2 (the modified kapsalis crossover and the original swap mutation) applied at the corresponding rates of 80% and 0.05% per individual. We also compared the 1-opt procedure proposed in Algorithm 7.4 with a proper simulated annealing algorithm run for a variable number of iterations. A similar idea was initially introduced in [55] in which a transmitter and its assigned frequency were selected among those violating at least one of the hard constraints and then being re-assigned a new frequency with the best alternative. Subsequently, this mutation was extended in [71] to all the transmitters and all the frequencies, thus generating a complete 1-opt. For these experiments we used a population of 20 individuals run for 2,000 generations (which with the use of the 1-opt procedure corresponds to $1,000,000 * |V|$ evaluations approximatively). Note that these values of the parameters will be kept for the rest of the experiments performed by this type of GA. In order to produce a given number of evaluations the number of generations performed will be suitably modified. Furthermore, when the decomposed approach is applied, the number of generations for each subset have been reduced proportionally to the number of subsets as required by Definition 4.1.

For the application of SA as a LS procedure, the mutation used (called *swap mutation*, see [68]), consists of a number of simple frequency swaps between pairs of transmitters selected at random, according to a given mutation rate. Finally, to improve the GA performance an *iterative 1-opt LS* procedure (see [40]) has been added after offspring generations to search for local optimality. the number of generations was set in order to roughly produce the same number of total evaluations. Results for Siemens2 are given in Table C.11. The outcomes produced by the 1-opt procedure appear competitive with SA and, consequently, this type of LS has been adopted in the rest of the experiments conducted for this thesis. 1-opt will be applied with a full rate of 100%, that is for each of the new individuals generated by the genetic operators. Finally, Tables C.12 and C.13 show, for Siemens1 and Siemens2 respectively, a comparison between the results produced with and without the addition of the LS to the direct GA applied with the same parameters described above. As will be described in more detail in the next section, these represent an example in which the decomposition approach is effective (Siemens1), and a second one in which it appears unsuccessful instead (Siemens2). Consequently, these outcomes should fairly reflect the contribution brought about by the LS during a run of the algorithm.

Note that, although the figures in Table C.11 generally improve with the amount of SA added, this cannot be further increased without distorting the natural structure of the GA. Moreover, since a heavy amount of SA also increases the computational complexity of the algorithm (thus limiting the runs to few generations) the GA assumes finally the structure of a genetic framework for the local search method rather than that of a proper evolutionary algorithm. Table C.14 shows some examples for Siemens1 and Siemens2 in which the GA incorporates the SA procedure, which is run each time for a very high number of iterations corresponding to $100,000*|V|$ and $1,000,000*|V|$ configurations explored. This 'memetic' GA used a population of only 10 individuals run for 10 generations. The results obtained are slightly better but essentially in line with those produced by different single runs of SA for the same number iterations (see A). This confirms that the 'memetic' GA is actually very similar to the local search method itself, thus losing the most of its evolutionary characteristics. For these reasons, this approach will not be further

Table C.11: Comparison of LSs for the COST259 Siemens2 benchmark

LS	iterationNo (SA)	Mean cost O_{MI}
-	-	21.53
1-opt	-	18.28
SA	100	19.98
1-opt + SA	100	17.94
SA	1,000	18.47
SA	10,000	17.40

Table C.12: Best and average cost O_{MI} (over three runs) for the COST259 Siemens1 benchmark solved with decomposition with and without the 1-opt LS

subsetsNo	LS 1 opt	no LS
1	4.14 (4.36)	4.99 (5.18)
2	3.79 (3.84)	4.61 (4.87)
4	4.05 (4.07)	4.85 (4.96)

used for the remaining experiments performed.

For the MI-FAP the order based representation has been used with the GA only for a number of comparison tests using the same NGSA-II framework already used for the direct representation. Here the tuning tests focussed on the choice of the crossover operator. The cycle crossover previously used in this thesis (see Section 5.1) has been compared in some preliminary experiments with a recently proposed operator specifically developed for the order based representation. This new operator, called *merge crossover* [11], has proven to be very successful in some tests performed on the similar graph coloring problem. It acts directly on

Table C.13: Best and average cost O_{MI} (over three runs) for the COST259 Siemens2 benchmark solved with decomposition with and without the 1-opt LS

subsetsNo	LS 1 opt	no LS
1	18.45 (18.72)	21.44 (21.53)
2	19.27 (19.38)	20.47 (21.04)
4	21.62 (21.02)	22.52 (23.81)

Table C.14: Memetic algorithm results for increasing percentages of SA used as LS for the COST259 Siemens1 and Siemens2 benchmarks

iterationNo (SA)	Siemens1	Siemens2
	Mean cost O_{MI}	
10,000	17.40	3.49
100,000	16.29	2.98
1,000,000	15.44	2.67

Table C.15: Performance of different crossovers for the COST259 Siemens1 benchmark solved by the order-based GA

Crossover	Mean cost O_{MI}
cycle cross.	5.27
merge cross.	5.03

permutations and begins merging two parents, as in a shuffle. Subsequently, it extracts two children from the merged list consisting of the lists of first or second instances of each values (see [11] for a detailed description of the procedure).

As an example, we show in Table C.15 the results obtained for Siemens1 by running the GA with the same parameters values adopted as a final choice for the direct representation, which correspond to $1,000,000 * |V|$ total evaluations approximatively as mentioned above.

The merge crossover produces slightly better results and, since the computational effort is actually the same as that required by the direct crossover, it has been chosen for the experiments conducted for the MI-FAP with the order-based representation. Note that, the cost produced by the order based GA is still about 20% worse than that produced by the direct representation. This tendency will be confirmed by other comparison results presented in Section 7.3.

Finally we report in the following two examples of cycle and merge crossover:

Cycle crossover

The cycle crossover operator identifies a number of cycles between two parent chromosomes (strings) represented by permutations of a given number of integers.

Parent 1: 8 4 7 3 6 2 5 1 9 0

Parent 2: 0 1 2 3 4 5 6 7 8 9

Cycles are built as follows:

For the first cycle we select a gene (i.e a value) at random in Parent 1. For simplicity in this example we start with the first value in Parent 1 (8) and select the corresponding position in Parent 2 which is 0.

Parent 1: **8** 4 7 3 6 2 5 1 9 0

Parent 2: **0** 1 2 3 4 5 6 7 8 9

We then have the first value for Offspring 1 which is 8 in the first position:

Offspring 1: 8 - - - - - - - - -

Then, we repeat the procedure starting from 0 in Parent 1, which is found at the 10th position where we select 9.

Parent 1: 8 4 7 3 6 2 5 1 9 **0**

Parent 2: 0 1 2 3 4 5 6 7 8 **9**

We then add the second value to Offspring 1, which is 0 in position 10th

Offspring 1: 8 - - - - - - - - 9

Again, we look for 9 in Parent 1 and find it in the 9th position then we select 8 in Parent 2. Because 8 was the value from which we started the cycle the procedure ends.

Parent 1: 8 4 7 3 6 2 5 1 **9** 0

Parent 2: 0 1 2 3 4 5 6 7 **8** 9

Similarly we add the last value 9 in position 9th to the offspring.

Offspring 1: 8 - - - - - - - 9 0

We now complete the values from offspring 1 by filling the remaining positions from string Parent 2 thus:

Offspring 1: 8 1 2 3 4 5 6 7 9 0

We can produce a second offspring as the 'complementary' of Offspring 1 defined as:

Offspring 2: 0 4 7 3 6 2 5 1 8 9

Mergecrossover

The merge crossover operator is presented in Figure C.4. Initially the two parents of size *n* are randomly merged into a single 2*n* element list (like in a card shuffle) Then each value in the merged list gives the ordering of the elements in the first offspring. The remaining value compose the second offspring with the same ordering they have in the merged list.

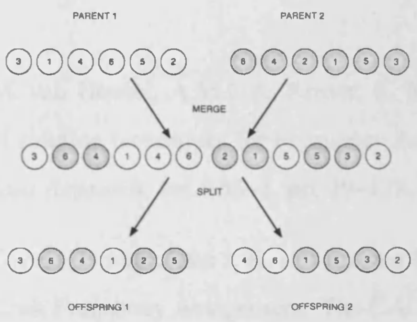


Figure C.4: Merge crossover example

More details about cycle and merge crossover can be found respectively in [118] and [90].

Bibliography

[1] *Cardiff University Condor Pool*, url="http://www.cardiff.ac.uk/insrv/it/condor/index.html", accessed on 1st July 2008,

[2] *FAP web – A website about Frequency Assignment Problems*, url = "http://fap.zib.de/", accessed on 1st July 2008.

[3] K.I. Aardal, S.P.M. van Hoesel, and B. Jansen, *A branch–and–cut algorithm for the frequency assignment problem*, R.M. 96\11, Maastricht University, 1996.

[4] K.I. Aardal, S.P.M. van Hoesel, A.M.C.A. Koster, C. Mannino, and A. Sassano, "Models and solution techniques for Frequency Assignment Problems", *Annals of Operations Research*, vol. 153–1, pp. 79–129, 2007.

[5] K. I. Aardal and C. A. J. Hurkens and J. K. Lenstra and S. R. Tiourine, "Algorithms for Radio Link Frequency Assignment: The CALMA Project", *Annals of Operations Research*, vol. 50–6, pp. 968–980, 2002.

[6] M. Alabau, L. Idoumghrar, and R. Schott, "New hybrid genetic algorithm for the frequency assignment problem", *IEEE Transactions on Broadcasting*, vol. 48–1, pp. 27–43, 2002.

[7] S.M. Allen, D.H. Smith, and S. Hurley, "Generation of lower bounds for minimum span frequency assignment" *Discrete Applied Mathematics*, vol. 119-1-2, pp. 59–78, 2002.

[8] S. M. Allen, D. H. Smith, and S. Hurley, "Lower bounding techniques for frequency assignment", *Discrete Mathematics*, vol 197–198, pp. 41–52, 1999.

[9] S.M. Allen, N. Dunkin, S. Hurley, and D.H. Smith, *Frequency assignment problems: benchmarks and lower bounds*, University of Glamorgan, 1998.

[10] L. G. Anderson, "A Simulation Study of some Dynamic Channel Assignment Algorithms in a High Capacity Mobile Telecommunications System", *IEEE Transactions on Communications*, vol. 21, pp. 1294–1301, 1973.

[11] P. G. Anderson, and D. Ashlock, "Advances in Ordered Greed", In *Proceedings of the Artificial Neural Networks In Engineering Conference (ANNIE 2004)*, Saint Louis, MO, USA, 2004.

[12] A. Avenali, C. Mannino, and A. Sassano, "Minimizing the span of d–walks to compute optimum frequency assignments", *Mathematical Programming*, vol. 91, pp. 357–374, 2002.

[13] D. Beckmann, and U. Killat, *Frequency Planning with respect to Interference Minimization in Cellular Radio Networks* COST 259 TD (99) 032, Vienna, Austria 1999.

[14] D. Beckmann, and U. Killat, "A new strategy for the application of genetic algorithms to the channel–assignment problem", in *IEEE Transactions on Vehicular Technology*, vol. 48, pp. 12611269, 1999.

[15] A. Bouju, J. F. Boyce, C. H. D. Dimitropoulos, G. vom Scheidt, and J. G. Taylor, "Tabu search for the radio links frequency assignment problem", *Proceedings of the UNICOM Conference on Applied Decision Technologies*, 1995.

[16] U. Brandes, M. Gaertler, and D. Wagner, "Experiments on Graph Clustering Algorithms", *Proceedings of the 11th Annual European Symposium on Algorithms (ESA'03)*, Budapest, 2003, pp. 568–579.

[17] D. Brelaz, "New methods to color the vertices of a graph", *Communications of the ACM*, vol. 22–4, pp. 251–256, 1979

[18] A. Capone, and M. Trubian, "Channel assignment problem in cellular networks: a new model and a tabu search algorithm",*IEEE Transactions on Vehicular Technology*, vol. 48-4, pp. 1252–1260, 1999.

[19] D. J. Castelino, S. Hurley, and N. M. Stephens, "A tabu search algorithm for frequency assignment", *Annals of Operations Research*, vol. 63, pp. 301–319, 1996.

[20] M. Chiarandini, and T. Stutzle, "Stochastic local search algorithms for Graph Set T-Coloring and Frequency Assignment", *Constraints*, vol. 12–3, pp. 371–403, 2007.

[21] R.H. Cheng, C.Y. Yu, and T.K. Wu, "A novel approach to the fixed channel assignment problem", *Journal of Information Science and Engineering*, vol. 21, pp. 39–58, 2005.

[22] C.A. Coello, "A Comprehensive Survey of Evolutionary–Based Multiobjective Optimization Techniques", In *Knowledge and Information Systems. An International Journal*, vol. 1–3, pp. 269–308, 1999.

[23] G. Colombo, and C.L. Mumford, "Comparing Algorithms, Representations and Operators for the Multi–objective Knapsack Problem", *Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC20052)*, Edinburgh, Scotland, 2005, pp. 1268–1275.

[24] G. Colombo, S. M. Allen, and R. M. Whitaker, "Genetic algorithms for large frequency assignment problems", *Proceedings of the 6th Annual PostGraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting*, Liverpool, UK, 2005.

[25] G. Colombo, “A Genetic Algorithm for frequency assignment with problem decomposition”, *International Journal of Mobile Network Design and Innovation*, vol. 1–2, pp. 102–112, 2006.

[26] G. Colombo, and S.M. Allen, “Problem Decomposition for Minimum Interference Frequency Assignment”, *Proc. of the 2005 IEEE Congress on Evolutionary Computation (CEC2007)*, Singapore, 2007.

[27] R. Carraghan, and P. M. Pardalos, “An exact algorithm for the maximum clique problem”, *Operations Research Letters*, vol. 9, pp. 375–382, 1990.

[28] L.M. Correia, Editor, *Wireless Flexible Personalised Communications*, Chichester, UK: Wiley Europe, 2001.

[29] C. Cotta, and J. M. Troya, “A comparison of several evolutionary heuristics for the frequency assignment problem”, In *Lecture Notes in Computer Science*, vol. 2084, pp. 709–716, Berlin Heidelberg, 2001.

[30] C. Crisan, and H. Muhlenbein, “The frequency assignment problem: a look at the performance of evolutionary search”, In *Selected Papers from the Third European Conference on Artificial Evolution*, pp. 363–274, 1997.

[31] C. Crisan, and H. Muhlenbein, “The breeder genetic algorithm for frequency assignment”, *Lecture Notes on Computer Science*, vol. 1498, pp. 897–906, 1998.

[32] L. D. Davis, “Handbook of Genetic Algorithms”, Van Nostrand Reinhold, New York, 1991.

[33] K. Deb, *Multi–Objective Optimization Using Evolutionary Algorithms*, Chichester, UK: Wiley Europe, 2002.

[34] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA–II”, *IEE Transactions on Evolutionary Computation*, vol. 6, pp. 182–197, 2002.

[35] K. Deb and D. A. Goldberg , "an investigation of niche and spieces formation in genetic function optimization", *Proceedings of the Third International Conference on Genetic Algorithms* , San Mateo, United States pp. 42–50, 1989.

[36] P.P Demestichas, E. C. Tzifa, M. E. Theologou, and M. E. Anagnostou. "Interference–oriented carrier assignment in wireless communications", *Communications Letters, IEEE* vol. 7, pp. 7–9, 2003.

[37] K.A. Dejong, *an analysis if the behaviour of a class of genetic adaptive systems*, University of Michigan Ann Arbor, MI, USA, 1975.

[38] N. W. Dunkin, and S. M. Allen, "Frequency assignment problems: representations and solutions", Technical Report CSDTR9714, June 1997.

[39] Eindhoven RLFAP Group, "Radio link frequency assignment project", Technical report, Eindhoven University of Technology, 1995.

[40] A. Eisenbl*ätter*, *Frequency Assignment in GSM Networks: Models, Heuristics, and Lower Bounds* PhD thesis, Technische Universitat Berlin, Germany, 2001.

[41] A. Farago, "Scalable analysis and design of ad hoc networks via random graph theory", In *DIALM 02: Proceedings of the 6th international workshop on Discrete algorithms and methods for mobile computing and communications*, New York, NY, USA, pp. 43-50, 2002.

[42] M. Fischetti, C. Lepschy, G. Minerva, G. Romanin–Jacur, and E. Toto, "Frequency assignment in mobile radio systems using branch–and–cut techniques", in *European Journal of Operational Research*, vol. 123, pp. 241–255, 2000.

[43] C.M. Fonseca and P. J. Fleming, "An overview of evolutionary algorithms in multiobjective optimization", In *Evolutionary Computation*, vol. 3–1, pp. 1–16, 1995.

[44] A. Gamst, "Some lower bounds for a class of frequency assignment problems", *IEEE Transactions on Vehicular Technology*, vol. 35, pp. 8–14, 1986.

[45] S.C. Ghosh, B.P. Sinha, and N. Das, "Channel assignment using genetic algorithm based on geometric symmetry", *IEEE Transactions on Vehicular Technology*, vol. 52, pp. 860–875, 2003.

[46] A. I. Giortzis and L. F. Turner, "A mathematical programming approach to the channel assignment problem in radio networks", *Proceedings of the IEEE 46th Vehicular Technology Conference* 1996.

[47] F. Glover, "Tabu search – part 1", *ORSA Journal on Computing*, vol. 1, pp. 190-206, 1989.

[48] D. A. Goldberg, and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization", *Proceedings of the Second International Conference on Genetic Algorithms and their application*, Cambridge, Massachusetts, United States, pp. 41–49, 1987.

[49] D. A. Goldberg, and D. E. Chang, "Tournament selection, niching, and the preservation of diversity", Technical report 91011, University of Illinois , USA, 1991.

[50] U. Gotzner, A. Gamst, and R. Rathgeber, "Statial Traffic Distribution in Cellular Networks", In: *Proc. IEEE VTC'97 Ottawa Canada*, pp. 1994–1998, 1997.

[51] D. Grace, A. G. Burr, and T. C. Tozer, "Comparison of Different Distributed Channel Assignment Algorithms for UFDMA", In *2nd IEEE International Conference on Personal, Mobile and Spread Spectrum Communications*, pp. 38–41, 1996.

[52] J.S. Graham, R. Montemanni, J.N.J. Moon, and D.H. Smith " frequeny assingment multple interference and binary constraints", *Wireless Networks*,

to appear, url="http://www.springerlink.com/content/ng41225q0855g2j2/", accessed on 1st July 2008.

[53] W. K. Hale, "Frequency assignment: Theory and applications", *Proc. IEEE*, vol. 38, pp. 1497-1514, 1980.

[54] W. K. Hale, "New spectrum management tools", *Proc. IEEE International Symposium on Electromagnetic Compatibility*, pp. 47-53, 1981.

[55] J. K. Hao, and R. Dorne, *Study of genetic search for the frequency assignment problem*, In *Selected Papers from the Third European Conference on Artificial Evolution*, pp. 333–334, 1997.

[56] M. Hata, "Empirical formula for propagation loss in land mobile radio services", *IEEE Transactions on Vehicular Technology*, vol. 29, 1980.

[57] M. Hellebrandt, and H. Heller, "A new heuristic method for frequency assignment", Tech. Report TD(00) 003, COST259, Valencia, Spain, Jan. 2000.

[58] M. J. H. Holland, *Adaptation in natural and artificial systems*, UMP, 1975.

[59] D. S. Hochbaum, "A new – old algorithm for minimum–cut and maximum–flow in closure graphs", *Networks*, vol. 37–4, pp. 171–193, 2001.

[60] S. Hurley, and D.H. Smith, "Meta–Heuristics and channel assignment", in *Methods and algorithms for radio channel assignment*, edited by S. Hurley and R. Leese, Oxford, UK: Oxford University Press, 2002.

[61] S. Hurley, R.M. Whitaker, and D. H. Smith, "Channel assignment in cellular networks without channel separation constraints", *Proceedings of IIIE vehicular technology conference fall*, 2000.

[62] S. Hurley, D. H. Smith, and S.U. Thiel, "Fasoft: a system for discrete channel frequency assignment", *Radio Science*, vol. 32–5 pp. 1921–1939, 1997.

[63] S. Hurley, D. H. Smith, and S.U. Thiel, "A comparison of local search algorithms for radio link frequency assignment problems", *Proceedings of the 1996 ACM symposium on Applied Computing*, pp. 251-257, 1996.

[64] S. Hurley, and D. H. Smith, "Fixed spectrum frequency assignment using natural algorithms", *Proceedings IEEE international conference on Genetic Algorithms in Engineering Systems*, pp. 373–378, 1995.

[65] S. Hurley, W. Crompton, and N. M. Stephens, "A parallel genetic algorithm for frequency assignment problems", In *Proceedings IMACS/IEEEInt. Symp. on Signal Processing, Robotics and Neural Networks*, pp. 81-84, Lille, France, 1994.

[66] C. Hurkens, and S. Tiourine, *Upper and lower bounding techniques for frequency assignment problems,* Technical report, Eindhoven University of Technology, 1995.

[67] J. Janssena, D. Krizancb, L. Narayananc, and S. Shended, "Distributed Online Frequency Assignment in Cellular Networks", *Journal of algorithms*, vol. 36–2 pp. 119–151, 2000.

[68] A. Kapsalis, P. Chardaire, V. J. Smith, and G. D. Smith, "The radio link frequency assignment problem: A case study using genetic algorithms", *Lecture Notes on Computer Science*, vol. 993, pp. 117–131, 1995.

[69] N. Karaoglu, and B. Manderick, "FAPSTER – a genetic algorithm for frequency assignment problem", *Proceedings of the 2005 Genetic and Evolutionary Computation Conference*, Washington, D.C., USA, 2005.

[70] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, "Optimization by simulated annealing", *Science*, vol. 220, pp. 671-680, 1983.

[71] A.W.J. Kolen, "A genetic algorithm for frequency assignment", *Statistica Neerlandica*, vol. 61–1, pp. 4–15, 2007.

[72] A. W. J. Kolen, C. P. M. van Hoesel, and R. van der Wal, *A constraint satisfaction approach to the radio link frequency assignment problem,* Technical Report 2.2.2, EUCLID CALMA project, 1994.

[73] M. Koppen and K. Franke, "Pareto-dominated hypervolume measure: an alternative approach to color morphology", *Proceedings of the Seventh International Conference on Hybrid Intelligent Systems*, pp. 234–239, 2007.

[74] A. M. C. A. Koster, C. P. M. van Hoesel, and A. W. J. Kolen, "The partial constraint satisfaction problem: Facets and lifting theorems", *Operations Research Letters*, vol. 23–35, pp. 89–97, 1998.

[75] A.M.C.A. Koster, C.P.M. van Hoesel, and A.W.J. Kolen, "Solving partial constraint satisfaction problems with tree decomposition", *Networks*, vol. 40–3, pp. 170–180, 2002.

[76] S.A. Kotrotsos, P.P Demestichas., E. C. Tzifa, M. E. Theologou, and M. E. Anagnostou, "A realistic interference–oriented version of the frequency planning problem for future wireless telecommunication systems", in *Proceedings of the Electrotechnical Conference MELECON 2000 10th Mediterranean*, pp. 311–314, 2000.

[77] T. Kurner, D.J. Chicon, and W. Wiesbeck, "Concepts and Results for 3D Digital Terrain–Based Wave Propagation Models: An Overview", *IEEE Journal on Selected Areas in Communications*, vol. 11–7, pp. 1002–1012, 1993.

[78] W. K. Lai, and G. Coghill, "Channel assignment through evolutionary optimization", *IEEE Transactions on Vehicular Technology*, vol. 45, pp. 91–96, 1996.

[79] T. L. Lau, *Guided Genetic Algorithm,* PhD thesis, University of Essex, 1999.

[80] C. Mannino, G. Oriolo, and F. Ricci, "The stable set problem and the thinness of a graph", *Operations Research Letters*, vol. 35, pp. 1–9, 2007

[81] C. Mannino, and A. Sassano, "An enumerative algorithm for the frequency assignment problem", *Discrete Applied Mathematics*, vol. 129–1, pp. 155–169, 2003.

[82] V. Maniezzo and R. Montemanni. *An exact algorithm for the min–interference frequency assignment problem.* Technical report, University of Bologna, 2000.

[83] S.Matsui, I. Watanabe, and K. Tokoro, "An efficient genetic algorithm for a fixed frequency assignment problem with limited bandwidth constraint", *Systems and Computers in Japan*, vol. 4, pp. 32-39, 2004.

[84] N. Metropolis, A. Rosenbluth, M. Rosenbluth, M. Teller, and E. Teller, "Equations of state calculations by fast computing machines", *Journal of Chemical Physics*, vol. 21, pp. 1087–1092, 1953.

[85] B. L. Miller and M. J. Shaw, "Genetic algorithms with dynamic niche sharing for multimodalfunction optimization", *Proceedings of IEEE International Conference on Evolutionary Computation* , Nagoya, Japan, pp. 786–791. 1996.

[86] R. Montemanni, D.H. Smith, and S.M Allen, "An ANTS algorithm for the minimum–span frequency–assignment problem with multiple interference", *IEE Trans. on Vehicular Technology*, vol. 51–5, pp. 949–953, 2002.

[87] R. Montemanni, J.N. Moon, and D.H. Smith, "An improved Tabu Search algorithm for the Fixed–Spectrum Frequency–Assignment problem", *IEE Trans. on Vehicular Technology*, vol. 52–3, pp. 891–901, 2003.

[88] R. Montemanni, D.H. Smith, and S.M. Allen, "Lower Bounds for Fixed Spectrum Frequency Assignment", in *Annals of Operations Research*, vol. 107–1, pp. 237–250, 2001.

[89] R. Montemanni, D.H. Smith, and S.M. Allen, "An improved algorithm to determine lower bounds for the fixed spectrum frequency assignment problem", in *European Journal of Operational Research.*, vol. 156, pp. 736–751, 2004.

[90] C.L. Mumford, "New Order-Based Crossovers for the Graph Coloring Problem ", *Parallel Problem Solving from Nature IX*, Reykjavik, Iceland, 2006.

[91] C. Y. Ngo, and V. O. K. Li, "Fixed channel assignment in cellular radio networks using a modified genetic algorithm", In *IEEE Transactions on Vehicular Technology*, vol. 47, pp. 163-171, 1998.

[92] A. Nolte, and R. Schrader, "Simulated Annealing and its Problems to Color Graphs", *Proceedings of the ESA 1996, Springer Lecture Notes in Computer Science*, pp. 137-151, 1996

[93] A. Nolte, and R. Schrader, "A note on the finite time behaviour of simulated annealing", *Operations Research Proceedings*, 1999.

[94] A. Nolte, and R. Schrader, "Simulated annealing and graph colouring", *Combinatorics, Probability and Computing*, vol. 10, pp. 29–40, 2001.

[95] P.R.J. Östergard, "A fast algorithm for the maximum clique problem", *Discrete Applied Mathematics*, vol. 120–1, 2002.

[96] P. Pardalos, J. Rappe, and M. Resende, "An exact parallel algorithm for the maximum clique problem", in *High Performance Algorithms and Software in Nonlinear Optimization*, edited by P.P.R. De Leone, A. Murl'i, G. Toraldo, Kluwer, Dordrecht, 1998.

[97] T. Park and C. Y. Lee, "Application of the graph coloring algorithm to the frequency assignment problem", *Journal of the Operations Research Society of Japan*, vol. 39, pp. 258–265, 1996.

[98] J. Peemoller, "A correction to Brelaz's modification of Brown's coloring algorithm", *Commun. ACM*, vol. 26–8, pp. 595–597, 1983.

[99] F. J. Romero, and D. M. Rodriguez, “Channel assignment in cellular systems using genetic algorithms”, In *Proceedings of the 46th IEEE Vehicular Technology Conference*, pp. 741–745, Atlanta, USA, 1996.

[100] S. Ruiz, X. Colet, J.J. Estevez, “Frequency planning optimisation in real mobile networks”, In *Proceedings of the 50th IEEE Vehicular Technology Conference*, vol. 4, pp 2082–2086, Amsterdam, Netherlands, 1999

[101] S. Salcedo–Sanz, and C. Bousoo–Calzn, “A hybrid neural–genetic algorithm for the frequency assignment problem in satellite communications”, *Applied Intelligence*, vol. 22–3, pp. 207-217, 2005.

[102] G. H. Sasaki, and B. Hajek, “The time complexity of maximum matching by simulated annealing”, *Journal of the Association of Computing Machinery*, vol. 35-2, pp. 387–403, 1998.

[103] M. Sevaux, and K. Saorenseny, “Permutation distance measures for memetic algorithms with population management”, In*Proceedings of the Sixth Metaheuristics International Conference*, Vienna, Austria, 2005;

[104] S. Shazely, H. Baraka, and A. Abdel–Wahab. “Solving graph partitioning problem using genetic algorithms”, In *Proceedings of the 1998 Midwest Symposium on Circuits and Systems*, pp. 302-305, 1998.

[105] Shin W.Y, S. Y. Chang, J. Lee, and C.H. Jun, “Frequency insertion strategy for channel assignment problem”, *Wireless Networks,* vol. 12, pp. 45-52, 2006.

[106] S. A. G. Shirazi, and M.B. Menhaj. “A new genetic based algorithm for channel assignment problems”, In *9th Fuzzy DaysInternational Conference on Computational Intelligence*, Dortmunt, 2006.

[107] K.A. Smith, “A genetic algorithm for the channel assignment problem”, *Proceedings Global Telecommunications Conference*, vol. 4, pp. 2013–2018, 1998.

[108] D.H. Smith, L.A. Hughes, J.N.J Moon, and R. Montemanni, “Measuring the effectiveness of frequency assignment algorithms” *IEEE Transactions on Vehicular Technology*, vol. 56-1, pp. 331–341, 2007.

[109] D.H. Smith, S. Hurley, and S.M. Allen, “A new lower bound for the channel assignment problem” *IEEE Transactions on Vehicular Technology*, vol. 49-4, pp. 1265–1272, 2000.

[110] D.H. Smith, S.M. Allen, S. Hurley, and W.J. Watkins, “Frequency Assignment: Methods and Algorithms”, in *NATO RTA SET/ISET Symposium on Frequency Assignment, Sharing and Conservation in Systems (Aerospace)*, Aalborg, Denmark, NATO RTO-MP-13, pp. K1–K18, 1998.

[111] D.H. Smith, S. Hurley, and S.U. Thiel, “Improving heuristics for the frequency assignment problem” *European Journal of Operational Reaearch*, vol. 107-1, pp. 76–86, 1998.

[112] D.H. Smith, and S. Hurley, “Bounds for the frequency assignment problem”, *Discrete Mathematics*, vol. 167–168, pp. 571–582, 1997.

[113] M. Stoer, and F.Wagner. “A simple min–cut algorithm”, In *Proceedings of the 2nd annual European symposium on algorithms*, pp. 141-147, 1994.

[114] DW. Tcha, Y. J. Chung, T. J. Choi, “A new lower bound for the frequency assignment problem”, *IEEE/ACM Transactions on Networking*, vol. 5, pp. 34–39, 1997.

[115] E. Tsang, and C. Voudouris, “Solving the radio link frequency assignment problem using guided local search”, in *NATO RTA SET/ISET Symposium on Frequency Assignment, Sharing and Conservation in Systems (Aerospace)*, Aalborg, Denmark, Paper 14b, 1998.

[116] A. Tsenov, "Simulated annealing and genetic algorithms in telecommunications network planning", *International Journal of Computational Intelligence*, vol. 2, pp. 240-245, 2005.

[117] L. J. de Urries, M. A. Diaz Guerra, and I. Berberana, "Frequency Planning using Simulated Annealing", Technical Report TD(00)–054, COST 259, 2000.

[118] C.L. Valenzuela, "A study of permutation operators for minimum span frequency assignment using an order based representation", *Journal of Heuristics*, vol. 7, pp. 5–21, 2000.

[119] C.L. Valenzuela, S. Hurley, and D. H. Smith, "A Permutation Based Genetic Algorithm for Minimum Span Frequency Assignment", *Lecture Notes In Computer Science*, vol. 1498, pp. 907–916, 1998.

[120] S. van Dongen, *A cluster algorithm for graphs*, Technical Report INS–R0010, National Research Institute for Mathematics and Computer Science in the Netherlands, Amsterdam, 2000.

[121] P. von Rickenbach, S. Schmid, R. Wattenhofer, and A. Zollinger, "A robust interference model for wireless ad–hoc networks", in *Proceedings of the 19th IEEE International Symposium on Parallel and Distributed Processing*, pp 239-249, 2005.

[122] S. Waharte, and R. Boutaba, *Comparison of Distributed Frequency Assignment Algorithms for Wireless Sensor Network*, Technical Report, University of Waterloo, ON, Canada.

[123] L. Wang, and W. Gu, "Genetic algorithms with stochastic ranking for optimal channel assignment in mobile communications", *Lecture Notes in Computer Science*, vol. 3314, pp. 154–159, 2004.

[124] C.L. Weston, S. Hurley, and R.M. Whitaker, "The effects of downlink transmission activity on service coverage in FDMA cellular networks", in *Proceed-*

ings of the 4th IASTED International multi–conference on wireless and optical communications conference, Banff, Canada, pp. 69–74, 2004.

[125] E. Zitzler, and L. Thiele, "Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach", In *IEEE Transactions on Evolutionary Computation*, vol. 3–4, pp. 257–271, 1999.