Optimizing Shortwave Radio Broadcast Resource Allocation via Pseudo-Boolean Constraint Solving and Local Search

Feifei Ma[1(✉)], Xin Gao[2], Minghao Yin[2], Linjie Pan[1], Jiwei Jin[3], Hai Liu[1], and Jian Zhang[1]

[1] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China
maff@ios.ac.cn, panlinjie1993@163.com, stuliuhai@gmail.com
[2] College of Computer Science, Northeast Normal University, Changchun, China
{gaox819,ymh}@nenu.edu.cn
[3] Shan Dong Jiaotong University, Jinan, China
jinjw@ios.ac.cn

Abstract. Shortwave radio broadcasting is the principal way for broadcasting of voice in many countries. An important problem in shortwave radio broadcasting is how to allocate transmission devices to radio programs, so that all radio programs are broadcasted properly and the overall broadcasting effect is optimized. The broadcasting effect of a program is determined by various factors, such as time, location, and device parameters. There are also restrictions on the usage of transmission devices. In this paper, we describe the allocation of shortwave radio broadcast resources as a constrained optimization problem and prove that it is NP-hard. A Pseudo-Boolean constraint formulation for the problem is presented. We also propose an efficient local search algorithm to solve the problem. Both methods are evaluated using real data. Experimental results suggest that we can find an allocation plan with good broadcasting effect quite efficiently.

1 Introduction

Shortwave radio is a kind of radio transmission using shortwave frequencies, ranging from 2 to 30 megahertz (MHz). Such radio waves can be reflected or refracted back to Earth from the ionosphere, allowing communication around the curve of the Earth. Therefore, shortwave radio is very effective for long distance communication. In many countries, shortwave radio broadcasting remains the principal way for broadcasting of voice and music.

An important problem arising in shortwave radio broadcasting is how to assign radio transmission resources to programs, so that the broadcasting quality is optimized. Generally, in a large country like China, there can be thousands of radio transmission devices distributed in dozens of shortwave radio stations all over the territory. For a radio program, the broadcasting effect in its target area may vary from one device to another. It is demanded that each radio program

M. Rueher (Ed.): CP 2016, LNCS 9892, pp. 650–665, 2016.
DOI: 10.1007/978-3-319-44953-1_41

should be transmitted with a proper transmission device so that the broadcasting satisfies certain criteria in the target area. Meanwhile, it is desirable to maximize the overall broadcasting effect.

In China, the state-level TV programs and radio programs are coordinated by the State Administration of Press, Publication, Radio, Film and Television (SAPPRFT[1]). Currently, the radio programs are managed by staff members manually. This is not efficient, and it is also error-prone. Since the search space of the problem is extremely huge, the staff members have no choice but to rely heavily on previous allocation plans, which are becoming obsolete with the change of programs and devices.

For the last few years, we have been cooperating with the Division of Radio Frequency Assignment of SAPPRFT, to increase the degree of automation in their daily work. Our aim is to design and implement a system that can automatically produce an optimal plan that allocates available transmission devices to the radio programs to be broadcasted. Basically, there are two kinds of tasks: the seasonal allocation and the daily allocation. They only differ in the number of the programs. Seasonal allocation needs to make arrangements for nearly a hundred programs, while daily allocation only handles several programs. So in this paper, we do not distinguish these two tasks. The challenges include, among others, (1) The optimal plan is to be made based on the broadcasting effect data of all possible allocations. Since there are thousands of devices and nearly a hundred radio programs, the search space of the problem is enormous. In fact, the raw data files produced by the propagation software for predicting broadcasting effect amount to hundreds of gigabytes (GB). (2) The problem is time-critical. Especially when some emergency occurs, the system should be able to adapt the allocation plan quickly so that all programs can be transmitted without pause.

The shortwave radio broadcast resource allocation problem (SRBRA) addressed in this paper is derived from our project. In order to focus on the algorithmic aspect, we omit in this paper the minor issues such as data processing and frequency selection which are actually laborious. We formulate shortwave radio broadcast resource allocation as a constrained optimization problem and prove that it is NP-hard. To solve the problem, we propose two complementary approaches, one is based on Pseudo-Boolean Optimization (PBO) and the other is an efficient local search algorithm with quick consistency checking mechanism and the metaheuristic of Greedy Randomized Adaptive Search Procedure [3,12].

The paper is organized as follows. We first describe the shortwave radio broadcast resource allocation problem and prove its NP-hardness in Sect. 2. In Sect. 3, we present a Pseudo-Boolean formulation for this problem. In Sect. 4 we describe the local search algorithm. We evaluate the proposed approaches on real data in Sect. 5. In Sect. 6 we discuss some related works. Finally we conclude the paper.

[1] http://www.sapprft.gov.cn/.

2 Problem Description

2.1 Background

Suppose there are m radio programs to be broadcasted by n radio transmission devices. We denote the ith $(1 \leq i \leq m)$ radio program by P_i and the jth $(1 \leq j \leq n)$ transmission device by D_j. We also denote the set of radio programs by $\mathcal{P}$ and that of transmission devices by $\mathcal{D}$. Program P_i is to be broadcasted to its target area R_i during a predetermined time span $[t_i, t'_i]$. In each target area, there are a number of monitoring sites. Note that two programs P_i and P_k may have the geographically same target area, but we still recognize R_i and R_k as separate areas in our problem. The reason is that P_i and P_k are broadcasted on different frequencies, so the broadcasting effect of one program would not interfere with that of the other. A transmission device consists of a transmitter and an antenna. Two different transmission devices D_j and D_k may share the same transmitter or antenna, or the same electric switch in their circuits, hence cannot be used simultaneously. Such devices are called **conflicting devices**, denoted by $conflict(D_j, D_k)$.

A program can only be transmitted with one device. The device D_j, once occupied by program P_i, cannot transmit other programs during the time span $[t_i, t'_i]$. Without loss of generality, we use $< P_i, D_j >$ to represent the allocation of transmission device D_j to program P_i. The broadcasting effect of $< P_i, D_j >$ at a monitoring site in R_i is measured by field strength and circuit reliability of the shortwave radio, both of which can be computed with a generic propagation program such as VOACAP [7] or REC533 [2]. The input to the propagation program includes the broadcasting time of P_i, parameters of D_j, the radio frequency, and the locations of the shortwave radio station and the monitoring site.

The broadcasting effect at a monitoring site is considered to be acceptable by the Division of Radio Frequency Assignment of SAPPRFT if the field strength is above 38 dB. The site is qualified if the field strength is above 55 dB and the circuit reliability is above 70 %. For an allocation $< P_i, D_j >$ to be admissible, if at least 60 % of the sites are acceptable. The optimization goal is to maximize the total number of qualified sites in the target areas of all programs.

To model an SRBRA problem, we need to gather a lot of information in advance. Data processing is an important module in our project, as is demonstrated in Fig. 1. As a preprocessing step, we employ a propagation program (REC533) to calculate the field strength and circuit reliability data for each allocation $< P_i, D_j >$ at every monitoring site in R_i, Then we sift out the admissible allocations. For each allocation $< P_i, D_j >$, the number of qualified sites in R_i, denoted as $N_{<P_i,D_j>}$, is also derived. Besides, for each program we find the best frequency to transmit according to certain criterion. In addition to broadcasting effect modeling, we also need to derive program information and device information. In particular, we generate the device conflicting constraints using a conflict checking algorithm. The information obtained from the data processing module is passed to the allocation algorithm, which is the main topic of this paper.

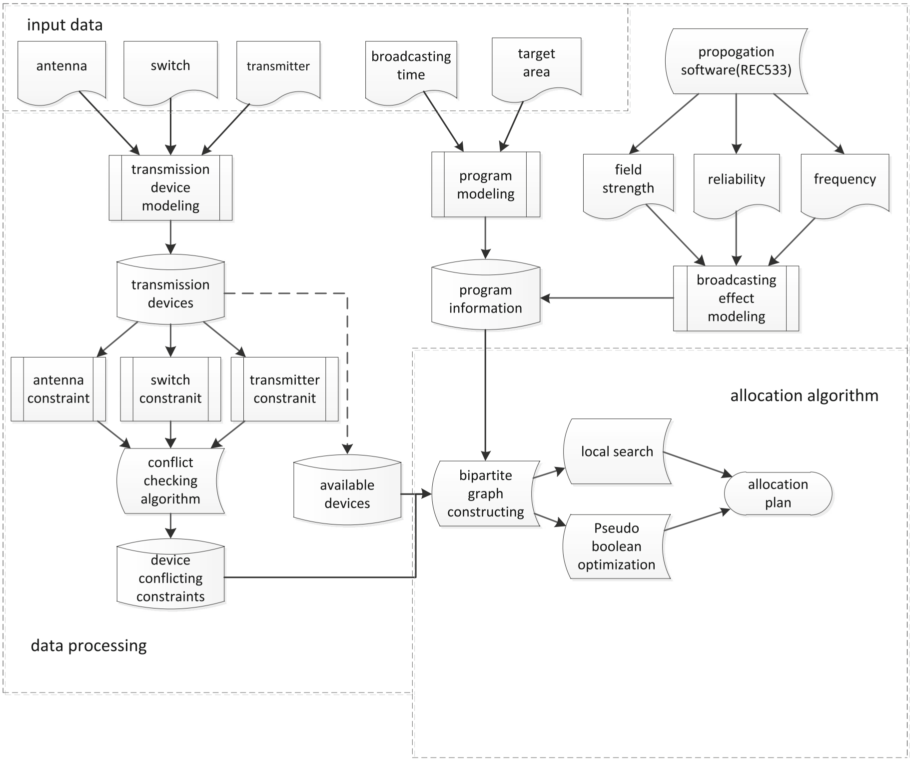

Fig. 1. The architecture of the SRBRA project

2.2 Shortwave Radio Broadcast Resource Allocation

Formally, the SRBRA problem can be defined in the following way.

Definition 1 (Shortwave Radio Broadcast Resource Allocation (SRBRA)). *Given m radio programs and n transmission devices, for each program P_i select a device D_j such that:*

1. *The allocation $< P_i, D_j >$ is admissible.*
2. *If program P_i and program P_k clash in the broadcasting time, i.e., $t_i < t'_k \wedge t_k < t'_i$, then they cannot be transmitted with the same device, or with two conflicting devices.*
3. *The total number of qualified sites $\Sigma_i N_{<P_i,D_j>}$ is maximized.*

The collection of the selected allocations is a solution to the problem, denoted by $\mathcal{S}$. The objective value corresponding to $\mathcal{S}$ is denoted by $N_{\mathcal{S}}$.

The constraint structure of an SRBRA problem can be represented as a variant of weighted bipartite graph, with $\mathcal{P}$ and $\mathcal{D}$ being two disjoint sets of vertices. If the allocation $< P_i, D_j >$ is admissible, there is an edge with weight

$N_{<P_i,D_j>}$ connecting the corresponding vertices. In illustration, we use a solid line to represent such an edge. Moreover, a dashed line is introduced for two programs clashing in the broadcasting time, or for two conflicting devices. The resulting graph is bipartite with respect to the solid lines.

Example 1. Figure 2 demonstrates the weighted bipartite graph of an SRBRA problem with 3 radio programs and 5 transmission devices. The number of qualified sites produced by each admissible allocation is labeled on the corresponding edge. The dashed line connecting P_2 and P_3 suggests that P_2 and P_3 clash in the broadcasting time. The dashed line between D_2 and D_3 suggests that they are conflicting devices. Similarly, we have $conflict(D_4, D_5)$. An optimal solution for this problem is $\{< P_1, D_2 >, < P_2, D_2 >, < P_3, D_5 >\}$. Accordingly, the optimal objective value is 23. Note that P_1 and P_2 can be transmitted with the same device because there is no clash in the broadcasting time.

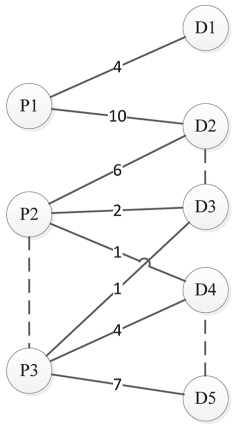

Fig. 2. An example of the SRBRA problem

At first glance, the SRBRA problem resembles the maximum weighted bipartite matching problem. The latter asks for the matching where the sum of the weights of the edges has a maximal value. There are two features that distinguish the SRBRA problem from maximum weighted bipartite matching which could be solved with Kuhn-Munkres algorithm [6,9] in polynomial time. Firstly, since each program has a fixed broadcasting time, a transmission device can be allocated to different programs if there is no clash in the broadcasting time. Hence the solution to the SRBRA problem is not a matching. Secondly, there are conflicts in the transmission resources, which means some edges cannot be selected simultaneously.

2.3 NP-hardness of SRBRA

We now prove the NP-hardness of the SRBRA problem.

Proposition 1. *The shortwave radio broadcast resource allocation problem is NP-hard.*

Proof. Consider the decision version of the SRBRA problem, which asks if it is possible to arrange m radio programs on n transmission devices so that the first two requirements in Definition 1 are satisfied. We can prove that this decision problem is NP-complete via reduction from the independent set problem. In graph theory, an independent set is a set of vertices in a graph, no two of which are adjacent. The problem of finding an independent set of certain size is a classical NP-complete problem in computer science. The general form of an independent set problem is: Given a graph G with n vertices, does there exist an independent set of size m ($m < n$)? We can construct a decision problem of SRBRA from the independent set problem by the following steps:

1. For each vertex v_j, create a transmission device D_j. If there is an edge connecting v_j and v_k in G, add a constraint $conflict(D_j, D_k)$.
2. Create m radio programs of identical broadcasting time. (The target areas can be arbitrary.)
3. Set the broadcasting effect of each allocation $< P_i, D_j >$ to be admissible, which means each program can be transmitted with all of the devices.

Apparently, the above procedure is a polynomial-time reduction. Suppose the SRBRA problem has a solution $\mathcal{S} = \{< P_i, D_j >\}$. The set of vertices corresponding to the devices that are allocated, denoted by $\mathcal{V} = \{v_j | \exists P_i < P_i, D_j > \in \mathcal{S}\}$, is a solution to the original independent set problem. This can be easily proved by reduction by contradiction. Firstly, the cardinality of $\mathcal{V}$ is m, otherwise, there must be at least one device allocated to two programs. Since the programs have the same broadcasting time, such allocations contradict the second requirement in Definition 1. Secondly, assume there are two adjacent vertices in $\mathcal{V}$, namely v_j and v_k. According to the construction, D_j and D_k are conflicting devices, and are used to transmit programs with the same broadcasting time. The second requirement in Definition 1 is violated too. Hence $\mathcal{V}$ is an independent set of size m. Conversely, if the independent set problem has a solution, the SRBRA problem is feasible, too.

Now that we have proved the NP-completeness of the decision version of the SRBRA problem, it easily follows that the problem itself is NP-hard. □

3 Pseudo-Boolean Formulation

The above problem can be naturally formulated as a Pseudo Boolean Optimization Problem. In its broadest sense, a Pseudo Boolean function is a function that maps Boolean values to a real number. However, in this context, we only need linear functions with integer coefficients to express the objective and the constraints. A linear Pseudo-Boolean Optimization Problem can be formally defined as:

$$Maximize \qquad \Sigma_j c_j x_j$$

$$Subject\ to \quad \bigwedge_i \Sigma_j a_{ij} x_j \leq b_i$$

where $x_j \in \{0, 1\}$ is a Boolean variable.

3.1 Encoding

For each admissible allocation, say $< P_i, D_j >$, we introduce a Boolean variable $A_{i,j}$ to indicate whether the allocation is adopted. In other words, $A_{i,j}$ is `true` if and only if P_i is transmitted with D_j.

Recall that our goal is to maximize the total number of qualified sites in the target areas of all programs. The objective function is as follows:

$$Maximize \quad \Sigma_i \Sigma_j N_{<P_i, D_j>} \times A_{i,j}$$

There are three types of linear Pseudo-Boolean constraints:

1. It is required that each program is transmitted with exactly one device. For each program P_i $(1 \leq i \leq m)$, we add the following Pseudo-Boolean constraint:
$$\Sigma_{j=1}^{n} A_{i,j} = 1 \tag{1}$$
2. If two programs P_i and P_k clash in the broadcasting time, then they cannot be transmitted with the same device. So for each pair of such programs we add the following constraints:
$$\bigwedge_{1 \leq j \leq n} A_{i,j} + A_{k,j} \leq 1 \tag{2}$$
3. We also have to prevent conflicting devices from being used simultaneously. Suppose P_i and P_k are two programs clashing in broadcasting time. A natural way to encode this restriction is as follows:
$$\bigwedge_{conflict(D_u, D_v)} A_{i,u} + A_{k,v} \leq 1 \wedge A_{i,v} + A_{k,u} \leq 1 \tag{3}$$
However, in our implementation, we find the following constraints more effective than Formula (3).
$$\bigwedge_{conflict(D_u, D_v)} A_{i,u} + A_{i,v} + A_{k,u} + A_{k,v} \leq 1 \tag{4}$$
It states that if D_u and D_v are conflicting devices, then only one of them can transmit one of the programs. Note that Formula (4) implies some constraints in Formula (2). But this kind of redundancy in encoding proved to be beneficial.

4 The Local Search Algorithm

The Greedy Randomized Adaptive Search Procedure (also known as GRASP) [3,12] is a metaheuristic algorithm commonly applied to combinatorial optimization problems. Generally, it consists of iterations over two phases: the greedy randomized construction of an initial solution and subsequent iterative improvements of the solution. Normally a solution to a combinatorial optimization problem is composed of many elements. In the construction phase, the solution is built by iteratively adding elements. Each element is randomly selected from a list of elements ranked by some greedy function according to the quality of the solution they will achieve. The list is called a restricted candidate list (RCL). During the improvement phase, the algorithm tries to improve the constructed solution by searching in its neighborhood. The two-phase process is executed repeatedly.

Our local search algorithm employs the GRASP metaheuristic. We first introduce several basic sub-procedures before describing the whole algorithm, which are devised taking into consideration the structure of the SRBRA problem.

4.1 Consistency Checking

The second requirement in Definition 1 imposes an important set of constraints. We should guarantee that during search the solution $\mathcal{S}$, partial or complete, always satisfies these constraints. Consistency checking is frequently invoked in our local search algorithm and its efficiency is critical to the success of the algorithm. If the consistency checking process is time-consuming, the local search algorithm can only visit a small portion of the whole search space, thus is likely to miss solutions with high quality. In the SRBRA problem, there are so many constraints, thus checking efficiently if any constraint is violated is nontrivial.

We devise the sub-procedure `Consistent(`$< P_i, D_j >$, $\mathcal{S}$`)`. It checks if the allocation $< P_i, D_j >$ is consistent with the current solution $\mathcal{S}$, as shown in Algorithm 1. Since each program is assigned only one device, the number of allocations in the current solution $\mathcal{S}$ is no more than the number of programs. Besides, in our implementation we use a two-dimensional matrix to store the conflicting relationship of devices, so $conflict(D_j, D_r)$ can be decided in constant time. The overall time complexity of `Consistent` is linear in the number of programs. Since the number of programs is quite small compared with the number of devices, the sub-procedure `Consistent` is very efficient.

Algorithm 1. Consistent($< P_i, D_j >$, $\mathcal{S}$)

for each $< P_k, D_r > \in \mathcal{S}$ **do**
  if $(conflict(D_j, D_r) \vee D_j = D_r) \wedge (t_i < t'_k \wedge t_k < t'_i)$ **then**
    return **false**;
  end if
end for
return **true**;

4.2 Greedy Randomized Construction

Algorithm 2 describes the greedy randomized construction procedure `Construct()`. At first, the solution $\mathcal{S}$ is initialized as an empty set. In each iteration, it randomly chooses a program P_i which has not been assigned a device. All devices are then examined for eligibility to transmit the program. There are two conditions for a device D_j to be eligible: Firstly, $< P_i, D_j >$ should be admissible; Secondly, the allocation $< P_i, D_j >$ should be consistent with the allocations already selected in the partial solution $\mathcal{S}$. The set of eligible devices is denoted by $\mathcal{C}$. We simply adopt the number of qualified sites as the greedy function. Therefore, the devices with the top 10 % highest numbers of qualified sites are selected into the set $\mathcal{C}^*$. Then P_i is assigned a device randomly chosen from $\mathcal{C}^*$.

Algorithm 2. Construct()

Solution $\mathcal{S} = \phi$;
repeat
  randomly choose an unassigned program $P_i \in \mathcal{P}$;
  $\mathcal{C} = \phi$;
  for each device $D_j \in \mathcal{D}$ **do**
    if $< P_i, D_j >$ is admissible and Consistent($< P_i, D_j >$,$\mathcal{S}$) **then**
      $\mathcal{C} = \mathcal{C} \cup \{D_j\}$;
    end if
  end for
  $\mathcal{C}^* = \{D_j | N_{<P_i,D_j>} is\ within\ the\ top\ 10\,\% in \mathcal{C}\}$;
  randomly choose a device D_r from $\mathcal{C}^*$;
  $\mathcal{S} = \mathcal{S} \cup \{< P_i, D_r >\}$;
until $\mathcal{P}$ is traversed.
return $\mathcal{S}$;

4.3 Operations in the Improvement Phase

We propose two operations that can locally improve a solution $\mathcal{S}$, `Swap(`$\mathcal{S}$`)` and `Substitute(`$\mathcal{S}$`)`.

The `Swap(`$\mathcal{S}$`)` operation illustrated in Algorithm 3 improves the solution $\mathcal{S}$ by selecting a pair of allocations in $\mathcal{S}$ and exchanging the corresponding devices. The allocation pair is greedily selected according to a rank function that evaluates the benefit of the exchange. For two allocations $< P_i, D_j >$ and $< P_k, D_r >$ in $\mathcal{S}$, swapping the devices would generate two new allocations $< P_i, D_r >$ and $< P_k, D_j >$. The new allocations should be both admissible and consistent with other allocations in $\mathcal{S}$. The resulting benefit is defined as the increment in the number of qualified sites, or formally:

$$score(\{< P_i, D_j >, < P_k, D_r >\}) = N_{<P_i,D_r>} - N_{<P_i,D_j>} + N_{<P_k,D_j>} - N_{<P_k,D_r>}$$

Finally, the pair of allocations with the highest score is swapped.

Algorithm 3. Swap($\mathcal{S}$)

$maxscore = 0$;
$bestpair = newpair = \phi$;
for any two allocations $< P_i, D_j >$ and $< P_k, D_r >$ in $\mathcal{S}$ **do**
  if $< P_i, D_r >$ and $< P_k, D_j >$ are both admissible **then**
    $pair = \{< P_i, D_j >, < P_k, D_r >\}$;
    if Consistent($< P_i, D_r >$, $\mathcal{S} \setminus pair$) and Consistent($< P_k, D_j >$, $\mathcal{S} \setminus pair$) **then**
      $score = N_{<P_i,D_r>} - N_{<P_i,D_j>} + N_{<P_k,D_j>} - N_{<P_k,D_r>}$;
      if $score > maxscore$ **then**
        $maxscore = score$;
        $bestpair = pair$;
        $newpair = \{< P_i, D_r >, < P_k, D_j >\}$;
      end if
    end if
  end if
end for
$\mathcal{S} = (\mathcal{S} \setminus bestpair) \cup newpair$;
return $\mathcal{S}$;

Algorithm 4. Substitute($\mathcal{S}$)

$maxscore = 0$;
$alloc1 = null, alloc2 = null$;
for Each allocation $< P_i, D_j > \in \mathcal{S}$ **do**
  for Each device $D_r \in (\mathcal{D} \setminus \{D_u | \exists P_k (< P_k, D_u > \in \mathcal{S})\})$ **do**
    if $< P_i, D_r >$ is admissible and Consistent($< P_i, D_r >$, $\mathcal{S} \setminus \{< P_i, D_j >\}$) **then**
      $score = N_{<P_i,D_r>} - N_{<P_i,D_j>}$;
      if $score > maxscore$ **then**
        $maxscore = score$;
        $alloc1 = < P_i, D_j >$;
        $alloc2 = < P_i, D_r >$;
      end if
    end if
  end for
end for
$\mathcal{S} = (\mathcal{S} \setminus \{alloc1\}) \cup \{alloc2\}$
return $\mathcal{S}$;

The `Swap` operation is a minor adjustment within the solution. The search procedure is likely to get stuck in the local optimum with respect to the neighborhood structure merely specified by `Swap`. Therefore, we introduce another operation `Substitute(`$\mathcal{S}$`)` which is described in Algorithm 4. The basic idea is to select an allocation $< P_i, D_j >$ in the current solution $\mathcal{S}$ and replace D_j with an idle device D_r. The selection is based on the greedy function evaluating the benefit of the substitution, which is defined as:

$$score(< P_i, D_j >, D_r) = N_{<P_i,D_r>} - N_{<P_i,D_j>}$$

The new allocation $< P_i, D_r >$ must be admissible and consistent with other allocations in $\mathcal{S}$, too. Obviously, the `Substitute` operation changes the set of allocated devices, thus may improve the quality of the solution.

4.4 The Local Search Procedure

The framework of our local search algorithm is shown in Algorithm 5, as described below. During the search, $\mathcal{S}^*$ keeps the best solution found thus far. In the beginning, an initial solution $\mathcal{S}$ is constructed, and the counters *swaps* and *substitutes* are set to 0. Then the algorithm iterates over the greedy randomized construction phase `Construct` and two improvement operations `Swap` and `Substitute` until the cut-off time is reached. There are two control parameters, I and R. At each step, if the current solution $\mathcal{S}$ is better than the best solution $\mathcal{S}^*$, the algorithm updates $\mathcal{S}^*$ and resets both of the counters. Otherwise, the `Swap` operation is firstly applied to improve the objective value $N_{\mathcal{S}}$. If `Swap` is applied up to I times and there is no improvement over $\mathcal{S}^*$, the `Substitute` operation is applied to improve solution $\mathcal{S}$, and the counter *swaps* is cleared. If $\mathcal{S}^*$ is still not improved by performing the `Substitute` operation R times, the algorithm restarts to get out of local optimum.

5 Empirical Evaluation

In this section we present experimental results of the proposed approaches on the set of real size shortwave radio broadcasting instances provided by the Division of Radio Frequency Assignment of SAPPRFT. These instances are taken from 7061 radio transmission devices and 87 programs. All instances are available on the website[2]. We use the solver `clasp`[3] for PBO solving. Since the PBO formulation of the SRBRA problem is naturally a 0-1 integer linear programming (ILP) problem, we also employ `CPLEX` as the ILP solver in the experiments.

We carried out two sets of experiments. The first one aims to compare the local search (LS) algorithm with the PBO approach and ILP solving, while the second one is mainly to evaluate the performance of the LS algorithm on large scale instances. The time limit for the LS algorithm is 10 seconds and for `clasp` and `CPLEX` is 3600 seconds. The number of qualified sites (denoted by #QS) obtained by each method is listed. Because of the randomness of the LS algorithm, it is executed 10 times for each instance, and both the maximum and average results are listed. Since `clasp` and `CPLEX` are exact solvers, if they terminate within the time limit, an optimal solution is found, and the corresponding result is marked with a star(*). Otherwise, we report the time they take to reach the primal solution at timeout. The times are measured in seconds. All experiments are performed on a computer with 3.3 GHz and 4 GB RAM under windows 7.

[2] http://ai.nenu.edu.cn/yinmh/.
[3] http://potassco.sourceforge.net/.

Algorithm 5. The Local Search Algorithm

```
Solution S* = S = Construct();
swaps = 0, substitutes = 0;
while elapsed time < cut-off time do
    if N_{S*} < N_S then
        S* = S;
        swaps = 0;
        substitutes = 0;
    else
        if (swaps < I) then
            swaps = swaps + 1;
            S = Swap(S);
        else
            if (substitutes < R) then
                substitutes = substitutes + 1;
                swaps = 0;
                S = Substitute(S);
            else
                S = Construct();
                swaps = 0;
                substitutes = 0;
            end if
        end if
    end if
end while
return S*;
```

Table 1 shows the results of comparing the LS algorithm with `clasp` and `CPLEX`. The number of devices is denoted by 'D" and the number of programs by "P". We can observe from Table 1 that `CPLEX` performs best. It finishes searching within one second for all instances, providing the optimal value. The LS algorithm produces a solution in less than one second for almost all instances, while `clasp` reaches the time limit for some instances. The LS algorithm can find the exactly optimal solution for many instances.

Table 2 shows the experimental results on large instances. `CPLEX` reports "out of memory" and does not provide any solution for all these instances, so we only list the results of `clasp` and the LS algorithm. The "total" is the total number of sites in the instance. The "rate" is cover rate, that is "#QS/total". The "UB" column gives the upper bound of #QS for each instance, which is obtained by simply selecting the best device for each program, ignoring the device conflicting constraints. The symbol '-' indicates that `clasp` fails to produce a solution at timeout. For those `clasp` does produce solutions, the results are unsatisfactory. By contrast, LS can finish all the instances within 10 seconds, with high cover rate meeting the need. The instance (D = 7061, P = 87) contains all usable official radio transmission devices and programs, and takes only 9.62 s with 81.1 % average cover rate. That means the LS algorithm can allocate all shortwave radio broadcast resource in China in a short time with high quality.

We also studied how the solution quality evolves as time passes for LS and `clasp` on two representative instances, as demonstrated in Fig. 3. The left picture indicates that for the instance with 400 devices and 20 programs, LS converges to the optimal solution much faster than `clasp` does. The picture on the right shows that the LS algorithm quickly reaches a good solution on the largest instance.

6 Related Works

The SRBRA problem is an extended resource allocation problem, which is a well-known problem widely studied in the literature. Numerous algorithms for variation of resource allocation problem are proposed in the decades. For example, for general resource allocation problem, [8] gives a hybrid search algorithm which combines genetic algorithm (GA) and ant colony optimization (ACO). Meanwhile special cases of resource allocation problem are studied widely. For example, [11] gives an algorithm for multi-vehicle systems with nonholonomic constraints. In this problem the vehicles satisfy a nonholonomic constraint, and the algorithm employs ideas from the traveling salesman problem and the path planning literature. [13] presents a proportional share resource allocation algorithm for real-time, time-shared systems.

Table 1. Comparison of `clasp`, `CPLEX` and LS

D	P	`clasp`		`CPLEX`		LS	
		#QS	time	#QS	time	max(avg) #QS	time
50	2	94*	<0.01	94*	<0.01	94(94)	<0.01
50	5	103*	0.171	103*	0.19	103(103)	<0.01
100	2	94*	0.016	94*	<0.01	94(94)	<0.01
100	5	103*	0.171	103*	0.14	103(103)	<0.01
200	2	121*	0.078	121*	0.14	121(121)	<0.01
200	5	238	7.269	238*	0.16	238(238)	<0.01
200	10	762	273.995	762*	0.27	762(762)	0.014
400	2	187*	0.577	187*	0.13	187(187)	<0.01
400	5	438	25.569	438*	0.09	438(438)	0.015
400	10	965	473.975	965*	0.19	965(965)	0.047
400	20	1228	3200.502	1232*	0.28	1232(1231.9)	3.411
800	2	212*	6.209	212*	0.16	212(212)	0.026
800	5	501	1589.128	501*	0.14	501(501)	0.127
2000	10	1305	1072.545	1555*	0.23	1555(1554.9)	2.667
3000	10	1393	1977.724	1562*	0.28	1562(1562)	1.284
4000	10	1246	1578.536	1677*	0.69	1677(1677)	1.006

Table 2. Experiments on large instances

D	P	clasp		LS			UB
		#QS	time	max(avg) #QS/total	max(avg) rate(%)	time	
2000	20	1830	2169.083	2183(2179.8)/2677	81.5(81.4)	4.180	2251
2000	30	710	3091.099	2464(2455.5)/3081	80.0(79.7)	3.260	2597
2000	40	1048	3177.101	3526(3484.2)/4618	76.4(75.4)	3.953	3858
2000	50	1020	1670.685	4180(4116.3)/5655	73.9(72.8)	3.510	4739
2000	60	847	991.647	4872(4806)/6687	72.9(71.9)	4.959	5619
2000	70	1182	2026.448	5707(5611.3)/8055	70.9(69.7)	6.137	6826
2000	87	-	-	6358(6234.2)/9404	67.6(66.3)	8.618	7981
3000	20	649	2286.309	2197(2192.4)/2677	82.1(81.9)	1.639	2251
3000	30	140	490.964	2502(2493.7)/3081	81.2(80.9)	2.528	2599
3000	40	540	377.552	3634(3591.1)/4618	78.7(77.8)	4.308	3860
3000	50	1014	1345.908	4306(4265.1)/5655	76.1(75.4)	5.971	4741
3000	60	1117	2112.735	5040(4996.8)/6687	75.4(74.7)	8.581	5623
3000	70	-	-	5969(5918)/8055	74.1(73.5)	9.934	6830
3000	87	-	-	6666(6609.7)/9404	70.9(70.3)	9.907	7990
4000	20	250	78.608	2320(2318)/2677	86.7(86.6)	2.716	2370
4000	30	115	853.837	2633(2626)/3081	85.5(85.2)	3.227	2753
4000	40	390	2998.060	3822(3800.8)/4618	82.8(82.3)	5.806	4092
4000	50	1060	3004.924	4577(4541.7)/5655	80.9(80.3)	8.500	5063
4000	60	-	-	5387(5329.4)/6687	80.6(79.7)	9.884	6040
4000	70	-	-	6284(6249.9)/8055	78.0(77.6)	9.806	7315
4000	87	-	-	7029(6986.5)/9404	74.7(74.3)	9.824	8570
5000	10	1182	2765.580	1682(1682)/1954	86.1(86.1)	1.312	1695
5000	20	320	2625.094	2338(2334.7)/2677	87.3(87.2)	2.493	2380
5000	30	514	76.035	2679(2660.7)/3081	87.0(86.4)	3.993	2763
5000	40	530	2324.841	3893(3877.9)/4618	84.3(84.0)	7.029	4103
5000	50	927	2892.373	4708(4681.5)/5655	83.3(82.8)	9.845	5077
5000	60	-	-	5496(5416.3)/6687	82.2(81.0)	9.912	6054
5000	70	-	-	6373(6318)/8055	79.1(78.4)	9.795	7335
5000	87	-	-	7186(7142.9)/9404	76.4(76.0)	9.712	8594
6000	10	719	148.395	1772(1772)/1954	90.7(90.7)	1.555	1783
6000	20	391	66.634	2456(2453.6)/2677	91.7(91.7)	2.805	2473
6000	30	411	199.213	2810(2804.3)/3081	91.2(91.0)	4.868	2861
6000	40	874	2329.879	4164(4137)/4618	90.2(89.6)	8.495	4307
6000	50	-	-	5030(4990.4)/5655	88.9(88.2)	9.803	5293
6000	60	-	-	5856(5788.9)/6687	87.6(86.6)	9.826	6288
6000	70	-	-	6878(6797.5)/8055	85.4(84.4)	9.842	7592
6000	87	-	-	7854(7667.2)/9404	83.5(81.5)	9.687	8875
7061	10	696	2734.533	1774(1774)/1954	90.8(90.8)	2.269	1785
7061	20	337	203.089	2458(2456.3)/2677	91.8(91.8)	3.358	2475
7061	30	249	2665.458	2812(2802.8)/3081	91.3(91.0)	5.796	2863
7061	40	629	1492.673	4192(4183.2)/4618	90.8(90.6)	9.778	4311
7061	50	-	-	5068(5045.9)/5655	89.6(89.2)	9.828	5297
7061	60	-	-	5883(5805.7)/6687	88.0(86.8)	9.909	6301
7061	70	-	-	6860(6800.8)/8055	85.2(84.4)	9.779	7605
7061	87	-	-	7739(7629.6)/9404	82.3(81.1)	9.620	8888

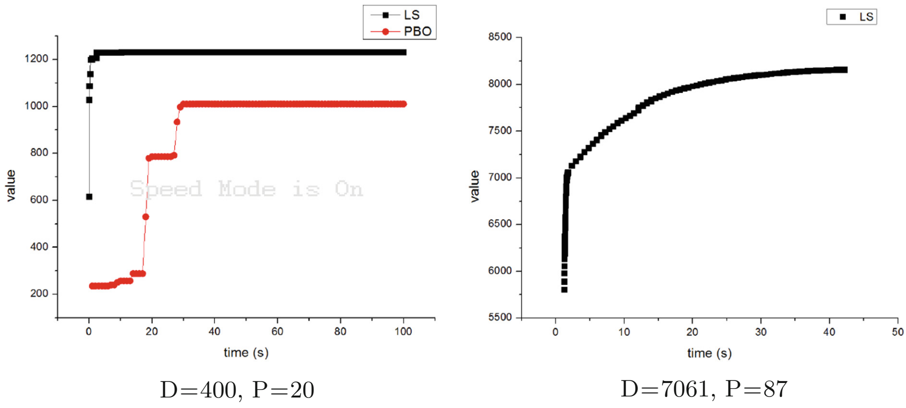

Fig. 3. Solution vs Time

For the case of radio, some effective algorithms are proposed to tackle the allocation and scheduling problem. [14] proposes a scheduling algorithm based on mean field annealing (MFA) neural networks for an optimal broadcast scheduling in packet radio networks. [10] also proposes a scheduling algorithm for multi-hop radio networks. [5] presents a radio resource allocation algorithm for Relay-Aided Cellular OFDMA System.

Some methods for the allocation and schedule problem for radio are based on CSP and Tabu Search. For example, [15] presents an algorithm for solving the frequency assignment problem (FAP) in cellular mobile systems, which uses CSP techniques. The algorithm represents a cell as a variable that has a very large domain, and determines a variable value step by step. [3] presents a tabu search algorithm for the FAP in mobile radio networks. [4] presents a modeling approach of the interference constraints and a probabilistic taboo search algorithm to solve the FAP in broadcasting. For more about FAPs, one can visit the website [1].

7 Conclusion

Shortwave radio broadcasting is still in heavy use in countries like China. In this paper, we studied the shortwave radio broadcast resource allocation problem (SRBRA), which is a kind of optimization problem with complex constraints. The SRBRA problem is derived from a real project in which we are cooperating with the Division of Radio Frequency Assignment of SAPPRFT. We have described the problem formally, and proved that it is an NP-hard problem. We proposed two complementary methods to solve SRBRA. One is to formulate it as a Pseudo-Boolean optimization (PBO) problem; and the other is a local search algorithm with quick consistency checking mechanism and the metaheuristic of Greedy Randomized Adaptive Search Procedure (GRASP). We have implemented both methods and evaluated them with real data from the Division of

Radio Frequency Assignment of SAPPRFT. It turns out that, we can find an allocation plan with good broadcasting effect quite efficiently.

References

1. Frequency Assignment Problem. http://fap.zib.de/
2. Hand, G.: VOACAP, ICEPAC and REC-533 propagation prediction programs for windows. *NTI/ITS*
3. Hao, J.-K., Dorne, R., Galinier, P.: Tabu search for frequency assignment in mobile radio networks. J. Heuristics **4**(1), 47–62 (1998)
4. Idoumghar, L., Debreux, P.: New modeling approach to the frequency assignment problem in broadcasting. IEEE Trans. Broadcast. **48**(4), 293–298 (2002)
5. Kaneko, M., Popovski, P.: Radio resource allocation algorithm for relay-aided cellular ofdma system. In: Proceedings of IEEE International Conference on Communications, pp. 4831–4836, June 2007
6. Kuhn, H.W.: The hungarian method for the assignment problem. Naval Res. Logistics Q. **2**(1–2), 83–97 (1955)
7. Lane, G.: Signal-to-Noise Predictions Using VOACAP-A Users Guide. Rockwell Collins Inc., USA (2001)
8. Lee, Z.-J., Lee, C.-Y.: A hybrid search algorithm with heuristics for resource allocation problem. Inf. Sci. **173**(1–3), 155–167 (2005)
9. Munkres, J.: Algorithms for the assignment and transportation problems. J. Soc. Ind. Appl. Math. **5**, 32–38 (1957)
10. Ramanathan, S., Lloyd, E.L.: Scheduling algorithms for multihop radio networks. IEEE/ACM Trans. Netw. **1**(2), 166–177 (1993)
11. Rathinam, S., Sengupta, R., Darbha, S.: A resource allocation algorithm for multivehicle systems with nonholonomic constraints. IEEE Trans. Autom. Sci. Eng. **4**(1), 98–104 (2007)
12. Resende, M.G.C., Ribeiro, C.C.: Greedy randomized adaptive search procedures: advances, hybridizations, and applications. In: Handbook of Metaheuristics, pp. 283–319. Springer, Boston (2010)
13. Stoica, I., Abdel-Wahab, H., Jeffay, K., Baruah, S.K., Gehrke, J.E., Plaxton, C.G.: A proportional share resource allocation algorithm for real-time, time-shared systems. In: Proceedings of IEEE Real-Time Systems Symposium, pp. 288–299, December 1996
14. Wang, G., Ansari, N.: Optimal broadcast scheduling in packet radio networks using mean field annealing. IEEE J. Sel. Areas Commun. **15**(2), 250–260 (1997)
15. Yokoo, M., Hirayama, K.: Frequency assignment for cellular mobile systems using constraint satisfaction techniques. In: Proceedings of IEEE Vehicular Technology Conference, vol. 2, pp. 888–894 (2000)