An Adaptive Neighborhood Search for k-Clustering Minimum Bi-clique Completion Problems

Mhand Hifi, Ibrahim Moussa, Toufik Saadi, and Sagvan Saleh

EPROAD EA 4669, Université de Picardie Jules Verne,
7 rue du Moulin Neuf, 80000 Amiens, France
{firstname.name}@u-picardie.fr

Abstract. In this paper, we propose to solve the k-clustering minimum bi-clique completion problem by using an adaptive neighborhood search. An instance of the problem is defined by a bipartite graph $G(V = (S,T),E)$, where V (resp. E) denotes the set of vertices (resp. edges) and the goal of the problem is to determine the partition of the set S of V into k clusters (disjoint subsets) such that the number of the edges that complete each cluster into a bi-clique, according to the vertices of T, should be minimized. The adaptive search is based upon three complementary steps: (i) a starting step that provides an initial solution by applying an adaptation of Johnson's principle, (ii) an intensification step in which both exchanging and k-opt strategies are introduced and, (iii) a diversification step that tries to explore unvisited solutions' space. The method is evaluated on benchmark instances taken from the literature, where the provided results are compared to those reached by recent methods available in the literature. The proposed method remains competitive and it yields new results.

Keywords: Bi-clique, combinatorial optimization, heuristic, local search.

1 Introduction

In this paper, we investigate the use of the neighborhood search for solving the so-called *k-Clustering minimum Bi-clique Completion Problem* (noted k-CmBCP). An instance of k-CmBCP is defined by a bipartite graph $G(V,E)$, where V is the set of n vertices such that $V = S \cup T$ and $S \cap T = \emptyset$ and, E denotes the set of edges. We recall that if (S,T) is a bi-clique graph, then each vertex of S (resp. T) is related to all the vertices of T (resp. S). Moreover, if the graph is formed with k bi-partite graphs (clusters), i.e., $\Big((S_1,T_1),(S_2,T_2),\ldots,(S_k,T_k)\Big)$, then all vertices of each couple (S_j,T_j), $j = 1,\ldots,k$, are interconnected. Because $G(V,E)$ is a general directed graph, looking for a k-clustering is equivalent to searching for a k bi-partite subgraphs of G with a minimum additional edges that do not belong to E. Similarly, the goal of the problem is to find the best partition of the set S into k clusters with a minimum additional edges.

As described in Gualandi *et al.* [4], such a problem has several real applications such as bundling channels for multicast transmissions. In such application, given a set of demands of services from customers, the aim consists of determining the k multicast sessions that is able to partition the set of the demands. Moreover, each of the considered service has to belong to a multicast session while each costumer can appear in several sessions.

The k-Clustering minimum Bi-clique Completion Problem (noted k-CmBCP) was first introduced by Faure *et al.* [2] in which the authors proved its NP-hardness. To our knowledge, very few paper dressing the KCmBCP are available in the literature. Among these papers, we cite the paper of Faure *et al.* [2] in which the authors proposed an integer linear programming model for solving small sized instances to optimality. In the same paper, a column generation-based heuristic has been also presented, where some large-scale instances of k-CmBCP have been tackled. Gualandi [3] tackled the problem by using a hybrid method; that is, an approach that combines constraint programming and semidefinite programming. Finally, Gualandi *et al.* [4] designed a special branch-and-price based method in order to accelerating the search process and improving the quality of the provided upper bounds when using the Cplex solver.

In this paper, we propose to solve the k-CmBCP by using an adaptive neighborhood search. Such an approach can be viewed as a variant of both methods proposed in Hifi and Michrafy [5] and a simplest version of Shaw's [7] large neighborhood search. We recall that an instance of k-CmBCP is characterized by a bipartite graph and the objective is to divide the first set S of vertices into k disjoint clusters, where the number of edges that must be added to form the k bi-cliques (representing the cost of the problem), should be minimum.

The rest of the paper is organized as follows. In Section 2, the k-CmBCP is first illustrated on an example and later its mathematical model is given. Section 3 discusses the tailored adaptive neighborhood search for approximately solving the k-CmBCP. The performance of the proposed algorithm is evaluated in Section 4, where its obtained results are compared to those reached by recent algorithms available in the literature. Finally, Section 5 summarizes the contribution of the paper.

2 The k-CmBCP

In this section, k-CmBCP is first illustrated throughout an example with $k = 2$. Second and last, the mathematical formulation of the k-CmBCP is given.

Fig. 1 illustrates a small example representing a k-CmBCP: the graph $G = (S, T)$ is defined by $S = \{1, ..., 4\}$ and $T = \{5, ..., 8\}$ and, the aim is to search the 2-clustering for G. First, a feasible solution (cf. Fig. 1.(a)) can be builded by partitioning the first set S as follows: $S_1 = \{1, 2\}$ and $S_2 = \{3, 4\}$, respectively. Second, because each partition forms a bipartite graph with its corresponding links belonging to the second set T, then it is necessary to add some connections between the cluster at hand and its corresponding links. In this case, Fig. 1.(b) shows, for both clusters, the added links (represented by the dashed edges for

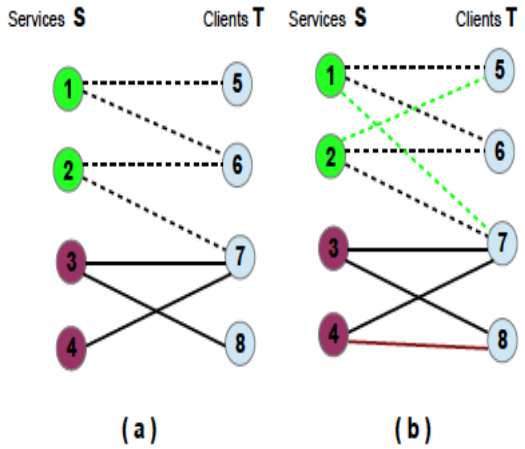

Fig. 1. Illustration of an instance of 2-CmBCP and its constructed feasible solution.

the first cluster S_1 and the bold edges for S_2) we need to make the solution feasible for k-CmBCP. Third and last, one can observe that the resulting cost is equal to 3, where the first cluster S_1 needs 2 additional edges and 1 for the second cluster S_2.

Formally the k-CmBCP can be stated as an Integer Linear Programming Model (ILPM):

$$\min \sum_{p \in K} \sum_{(i,j) \in \overline{E}} z_{ijp} \tag{1}$$

$$\text{s.t.} \tag{2}$$

$$x_{ip} + y_{jp} \leq 1 + z_{ijp}, \ \forall (i,j) \in \overline{E}, \ \forall p \in K \tag{3}$$

$$\sum_{p \in K} x_{ip} = 1; \ \forall i \in S \tag{4}$$

$$x_{ip} \leq y_{jp}, \ \forall (i,j) \in E, \ \forall p \in K \tag{5}$$

$$x_{ip}, y_{jp} \in \{0,1\}, \ \forall i \in S, \ \forall j \in T, \ \forall p \in K \tag{6}$$

$$z_{ijp} \in \{0,1\}, \forall (i,j) \in \overline{E}, \forall p \in K \tag{7}$$

where the objective function Equation (1) minimizes the number of edges that completes each induced bipartite subgraph into a biclique, in others words, it minimizes the total cost of all clusters (i.e., the number of edges added in this case). Equation (3) ensures the link between both variables x_{ip} and y_{jp} by setting z_{ijp} to 1 whenever the other two variables are fixed to 1. Equation (4) ensures that each vertex belonging to the set S should be assigned to a single cluster. Equation (5) forces each vertex $j \in T$, which is adjacent to another vertex $i \in S$ and assigned to the p^{th} cluster, to be affected to the same cluster (Of course,

we recall that some vertices can be assigned to several clusters). Finally, Equations (6) and (7) ensure the integrality of the variables. Note that, the limit of the formulation described above (equations (1) to (7)) is that any permutation of the indices p provide s the same optimal solution. The aforementioned issue was tackled by Fahle [1] by introducing the so-called symmetry-breaking constraints.

3 An adaptive neighbor search for k-CmBCP

In this section, we discuss the principle of the proposed adaptive neighborhood search for approximately solving the k-CmBCP. The used approach is mainly based upon three complementary steps. Such an approach has been already used in Hifi and Michrafy [5] for solving some knapsack type problems. It has also used by Shaw [7], where the author proposed a general purpose.

Herein, an adaptive method is considered, where the first step is applied in order to build a quick starting feasible solution. The second step tries to improve the quality of the starting solution by using the so-called *intensification strategy*. The aforementioned strategy alternatively combines both exchanging and 2-opt procedures. The third step introduces a *diversification strategy* in order to handle some sub-spaces with an efficient manner: both degrading and re-optimizing strategies are introduced for searching next feasible spaces.

3.1 A starting solution for k-CmBCP

There exists several ways for defining a starting solution when using heuristics. Herein, a simple greedy procedure is considered: it is based on the principle of Johnson's algorithm (cf., Johnson [6]). Indeed, in a bipartite graph, which can be considered as a special case of the set covering problem, an adaptation of Johnson's algorithm may be described as follows:

(i) Set $S' = S$,
(ii) Order all vertices of the first set S' in decreasing order of their degrees,
(iii) Select a vertex of S' having the greatest degree and put it into the current cluster (or into a new cluster to open),
(iv) Define a reduced graph G' by removing the last vertex from S',
(v) Stop if $S' = \emptyset$, repeat steps (ii)-(iv) otherwise.

Of course, the above procedure is used in order to partition the set S of the bipartite graph G. Then, in order to provide the cost of the starting solution reached by the greedy procedure, an adding step is used in order to complete each created cluster with the additional edges that forms the k-bicliques for G.

3.2 Improving the quality of the solutions: intensification step

Generally, improving a solution at hand requires to introduce a local search, where its aim is to perform an interesting search on a series of neighborhoods. Herein, the used intensification strategy is is based on combining both *exchanging*

strategy and a *special k-opt procedure*. The exchanging is employed between some services (vertices) of the current clusters (forming the set S), whereas the k-opt procedure considers several moves between clusters.

Algorithm 1 Improving the quality of the current solution.

Require: A starting feasible solution S with its added edges (noted e_{nb}).
Ensure: A (new) feasible solution S'.

1: $ok \leftarrow$ `false`;
2: **while** $\big($not(ok) and (a local stopping condition is not met)$\big)$ **do**
3: **for all** $k_j \in K$ **do**
4: **for all** $s_i \in k_j$ **do**
5: **for all** $k_\ell \in K$, $\ell \neq j$, **do**
6: Put s_i (from k_j) in k_ℓ;
7: Let S' be a new solution with its corresponding objective function e^{ℓ}_{nb};
8: **if** ($e^{\ell}_{nb} < e_{nb}$) **then**
9: $S \leftarrow S'$ and $ok =$`true`;
 else
10: Exit with the best solution of the neighborhood S^*, if a local stopping condition is met.
11: **end if**
12: **end for**
13: **end for**
14: **end for**
15: **end while**

Algorithm 1 describes the main steps of the intensification procedure which is used for improving the quality of the solutions at hand. Indeed, the input of the algorithm is the solution provided by the greedy procedure (with its objective value e_{nb} that measures the number of edges added to complete the solution) described in Section 3.1. Second, the main loop `while` (line 2) is used for stopping the search whenever the local search is able to provide a new feasible solution with a better value (in this case, a boolean, namely *ok*, is introduced in order to stop the loop). Of course, the stopping criteria is also consolidated with a runtime limit; that is, a runtime that the loop can take for exploring a series of neighborhoods (this parameter is experimentally defined). Third, an exchanging step is considered in lines from 3 to 6: in this case, two cases can be considered:

1. A vertex belonging to a cluster can be exchanged with another vertex that belongs to another one; the solution is updated when the new solution improves the quality of the best solution of the neighborhood.
2. When the previous step fails to provide a better solution, then a 2-opt procedure is introduced as follows: (a) exchange two vertices from a selected cluster and move them to other clusters (when it is possible) and, (b) recombine the clusters for providing a k-CmBCP's feasible solution.

Fourth and last, k-CmBCP's feasible solution is updated (line 9) if the current objective function improves the quality of the solution in the visited neighborhood.

3.3 Applying a diversification step

Now we are going to show how to simulate the diversification principle for searching a series of new solutions. Indeed, the intensification search can improve the solutions when it explores a series of neighborhoods, or when these neighborhoods are very close. Moreover, changing the space search induces, in some cases, the exploration of new neighborhoods and so, some new and better solutions may emerge. Herein, an adaptive neighborhood search is introduced, which is based upon *degrading strategy* and *re-optimization strategy* (as used in Hifi and Michrafy [5]). After applying both strategies, the intensification step (cf. Algorithm 1) is recalled in order to improve the quality of each solution at hand.

Now we are going to show how to combine both strategies for searching a series of new solutions. Let F be the current feasible solution obtained at the first step (or at internal of the iterative search). Let α be a constant, such that $\alpha \in [0, 100]$, denoting the percentage of free vertices of the set S, i.e., some vertices are free according to the current solution F . Then, the diversification procedure can be considered as an alternate approach which is performed by calling the following two procedures:

- *Degrading procedure*: Consider $F^{(\alpha)}$ as the free vertices' set associated to the current solution F, where $\alpha\%$ of vertices belonging to S are unassigned. Let $\overline{S}^{(\alpha)}$ be the complementary set of $S^{(\alpha)}$ according to the set of services S.
- *Re-optimization procedure*: Complete the current partial solution by using Johnson's algorithm on the subset $\overline{S}^{(\alpha)}$.

Note that, on the one hand, both degrading and re-optimizing procedures are combined in order to produce a series of partial feasible solutions of k-CmBCP and their improved ones. On the other hand, at each time, one can re-call Algorithm 1 in order to improve the quality of the solutions.

The main steps of such a procedure is detailed in Algorithm 2; that is, a process trying to simulate the diversification principle when varying both degrading and re-optimizing procedures. First, Algorithm 2 starts by setting the initial solution equals to the solution obtained by Algorithm 1 (of course, the provided solution can also reached at each internal search). The main loop (line 2: it is used to stop the search when the stopping criteria is performed; herein, a maximum number of iterations is considered) describes the principe of the diversification procedure when combining both degrading and re-optimizing strategies. In this case, it always starts with the best solution obtained up to now (initially, the solution is that reached by Algorithm 1). Next, the degrading procedure is called (line 6) in a local loop for randomly removing $\alpha\%$ of the vertices from the

Algorithm 2 An adaptive neighborhood search

Require: A starting feasible solution S_{start} of value $V(S_{start})$
Ensure: An approximate solution S_{best} of value $V(S_{best})$

1: $S_{best} = S_{start}$;
2: **while** (a stopping condition is not met) **do**
3: $S' = S_{best}$;
4: **while** (a local stopping condition is not met) **do**
5: $S_{local} = S'$;
6: Apply the `Removing strategy` on the current solution S_{local};
7: Call the `Re-optimizing strategy` for completing the current solution;
8: Let S' be the new obtained solution;
9: Call Algorithm 1 for improving the current solution S';
10: **If** $(V(S' < V(S_{best})))$, **then** $S_{best} = S'$;
11: **end while**
12: **end while**

current solution and trying to improve the quality of the loc al solution by the re-optimization procedure (line 7). The local loop serves to search the best solution on a series of neighborhoods an so, a small number of degrading / re-optimizing are introduced (herein, ten degrading / removing calls are considered). At each step of the local loop, the new provided solution is improved by calling Algorithm 1 (lines 8 and 9). Finally, the process is repeated with the best solution found to far, where the algorithm stops with the best k-CmBCP's approximate feasible solution when the maximum number of iterations is performed.

4 Computational results

This section investigates the effectiveness of the proposed Adaptive Neighborhood Search (noted ANS) on some instances extracted from Gualandi *et al.* [4]. Preliminary results obtained by the proposed ANS are compared to the best results available in the literature and to those reached by the ILPM (cf. Section 2) when solved with the Cplex solver [8] version 12.5 (limited to one hour and with the default settings). The proposed ANS was coded in C++ and tested on Pentium Core Duo 2.9 Ghz

Table 1 describes the main characteristics of the set of instances tested. Column 1 of the table indicates the instance label. Columns from 2 to 4 display cardinalities of both sets S (services) and T (costumers) and the number of clustering k needed. Finally, column 5 tallies the density (d) of the graph associated to each instance. Note also that these instances represent six group of instances, where each group is composed of three available instances and it depends on the variation of the number of services ($|S|$), costumers ($|T|$), the clusters (k) and the density d of its corresponding graph.

Inst.	S	T	k	d
I1.1	50	50	5	0.3
I1.2	50	50	5	0.5
I1.3	50	50	5	0.7
I2.1	50	50	10	0.3
I2.2	50	50	10	0.5
I2.3	50	50	10	0.7
I3.1	80	80	5	0.3
I3.2	80	80	5	0.5
I3.3	80	80	5	0.7
I4.1	80	80	10	0.3
I4.2	80	80	10	0.5
I4.3	80	80	10	0.7
I5.1	100	100	5	0.3
I5.2	100	100	5	0.5
I5.3	100	100	5	0.7
I6.1	100	100	10	0.3
I6.2	100	100	10	0.5
I6.3	100	100	10	0.7

Table 1. Description of the instances' characteristics.

4.1 Parameter settings

In the preliminary results, two main parameters should be taken into account by ANS (cf., Algorithm 2): the stopping criteria and the percentage of the removed vertices belonging to the current solution. Of course, in order to maintain the diversity of the solutions, we set the local stopping condition to twenty, which means that Algorithm 2 exists either by an improved solution reached before attaining the maximum number of iterations fixed or by the best solution reached in the neighborhood.

Because the improved procedure (cf., Algorithm 1) works when each solution is submitted to degradation and re-optimization, then the maximum number of global iterations was fixed to fifty. Finally, three values for the parameters α were considered: $\alpha \in \{2\%;\ 5\%;\ 10\%\}$. On the one hand, limited computational results showed that the algorithm with the value $\alpha = 2\%$ requires more runtime for improving some instances of the literature. On the other hand, when the algorithm is run with $\alpha = 10\%$, the obtained results becomes, in some cases, worse than those published in the literature. Only for $\alpha = 5$ (or closest to the aforementioned value) the algorithm seems to get more interesting results. Hence, the choice with $\alpha = 5$ is adopted for the rest of this paper.

Table 2 displays the variation of *Av_Sol* representing the average value of all solutions realized by ANS over all treated instances by fixing the same average runtime (1000 seconds, on average). It shows the variation of *Av_Sol* denoting the average value of solutions reached by ANS over all treated instances: the first column tallies the average results with $\alpha = 2\%$ and the third column displays

	Variation of α	
#Inst	2%	5%
I1.1	1319	1319
I1.2	1078	1072
I1.3	672	671
I2.1	938	938
I2.2	876	875,4
I2.3	577	575
I3.1	3820.2	3815
I3.2	2862	2862
I3.3	1769.2	1768.4
I4.1	3202	3202
I4.2	2571	2571
I4.3	1618.4	1618
I5.1	6248	6248
I5.2	4655.6	4650
I5.3	2842	2842
I6.1	4298	4298
I6.2	4298.4	5427.2
I6.3	2644	2644
Av_Sol	2644.4	2633.11

Table 2. Effect of α on the quality of the final solutions.

the average results for $\alpha = 5\%$. One can observe, globally for the same runtime, the average quality of the solutions realized is better when fixing α to 5%.

4.2 Effect of the degrading and re-optimizing procedures

This section evaluates the effect of the proposed method based upon degrading and re-optimizing strategies. Recall that the method works as follows. First, it deteriorates the starting / current solution by removing $\alpha\%$ of the variables fixed to one. Second, it re-optimizes the partial solution reached by using a greedy procedure and improving it using an intensification search. Third and last, it re-iterates the same resolution on the last solution until either no better solution is obtained or a maximum number of iterations is performed.

Table 3 shows the results realized by ANS and those extracted from Gualandi *et al.* [4] (noted GMM) and the solutions given by the Cplex solver 12.5 (of course, because the Cplex is tailored for solving the problems to optimality, then the runtime limit was extended to one hour). Column 1 represents the instance label. Column 2 displays the solution value reached by the Cplex solver and column 3 tallies Gualandi *et al.*s's solution values (representing the best solutions of the literature). Columns from 4 to 8 display the five solutions provided by ANS representing the five trials. Finally, columns 9 and 10 show the average solution values over the five trials and the best solutions reached by ANS over

			ANS's solution values						
#Inst	Cplex V_{Cplex}	GMM V_{GMM}	1	2	3	4	5	Av_Sol	Best
I1.1	1423	1321	1319	1319	1319	1319	1319	1319	1319
I1.2	1091	1072	1072	1072	1072	1072	1072	1072	1072
I1.3	676	672	671	671	671	671	671	671	671
I2.1	1135	938	938	938	938	938	938	938	938
I2.2	930	876	875	876	876	875	875	875.4	875
I2.3	587	577	575	575	575	575	575	575	575
I3.1	4166	3819	3815	3815	3815	3815	3815	3815	3815
I3.2	2931	2862	2862	2862	2862	2862	2862	2862	2862
I3.3	1775	1769	1768	1769	1768	1768	1769	1768.4	1768
I4.1	3735	3202	3202	3202	3202	3202	3202	3202	3202
I4.2	2675	2571	2571	2571	2571	2571	2571	2571	2571
I4.3	1634	1618	1618	1618	1618	1618	1618	1618	1618
I5.1	6645	6248	6248	6248	6248	6248	6248	6248	6248
I5.2	4716	4658	4650	4650	4650	4650	4650	4650	4650
I5.3	2854	2842	2842	2842	2842	2842	2842	2842	2842
I6.1	6119	4298	4298	4298	4298	4298	4298	4298	4298
I6.2	9111	5428	5427	5428	5427	5427	5427	5427.2	5427
I6.3	2704	2644	2644	2644	2644	2644	2644	2644	2644
Average	3050.39	2634.17	2633.06	2633.22	2633.11	2633.06	2633.11	2633.11	2633.06

Table 3. Performance of ANS vs Cplex and GMM algorithm.

these five trials.

The analysis of Table 3 follows.

1. First, one can observe the inferiority of Cplex solver since it is not able to match the best solutions of the literature.
2. Second, the best method of the literature (GMM) realizes 10 best solutions out of 18, representing a percentage of 55.56% of the best solutions. In fact, GMM matches all the ten instances, but there is no value better than the new solutions produced by ANS (cf., the last column of Table 3).
3. Third, among the five trials realized by ANS, one can observe that the average values (see the last line of Table 3) are better than GMM's average value. Indeed, ANS is able to realize an average value varying from 2633.22 (trial 2) and 2633.06 (trials 1 and 4) whereas GMM realizes an average value of 2634.17.
4. Fourth and last, regarding the best solutions realized by ANS (cf., column 10), one can observe that ANS is able to reach eight new solution values, which represents a percentage of more than 44% of the tested instances.

5 Conclusion

In this paper, an adaptive neighborhood search was proposed for approximately solving the k-clustering minimum bi-clique completion problem. The method is based upon degrading and re-optimizing strategies. The first strategy is used in order to diversify the search process and the second one is employed for repairing the configuration at hand. Both strategies were completed by an intensification procedure; that is, a neighboring search which tries to improve the quality of a series of provided solutions. Computational results showed that the proposed algorithm performed better than both Cplex solver and more recent method available in the literature by yielding high-quality solutions: it improved most than 44% of the best solutions of the literature.

References

1. Fahle, T.: Simple and Fast: Improving a Branch-and-Bound Algorithm for Maximum Clique. In: Mohring, R., Raman, R. (eds.) Algorithms – ESA 2002. LNCS, vol. 2461, pp. 485–498. Springer Berlin Heidelberg (2002)
2. Faure, N., Chrétienne, P., Gourdin, E., Sourd, F.: Biclique Completion Problems for Multicast Network Design. Discrete Optimization, 4, pp. 360–377 (2007)
3. Gualandi, S.: k-Clustering Minimum Biclique Completion via a Hybrid CP and SDP Approach. In: van Hoeve, W-J., Hooker, J.N. (eds.) Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. LNCS, vol. 5547, pp. 87–101. Springer Berlin Heidelberg (2009)
4. Gualandi, S., Maffioli, F., Magni, C.: A Branch-and-Price Approach to k-Clustering Minimum Biclique Completion Problem. International Transactions in Operational Research, 20, pp. 101–117 (2013)
5. Hifi, M., Michrafy, M.: A Reactive Local Search-Based Algorithm for the Disjunctively Knapsack Problem. Journal of the Operational Research Society, 57, pp. 718–726 (2006)
6. Johnson, D.S.: Approximation Algorithms for Combinatorial Problems. Journal of Computer and System Sciences, 9, pp. 256-278 (1974)
7. Shaw, P.: Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. In: M. Maher, M., Puget, J.F. (eds.) Principles and Practice of Constraint Programming – CP98. LNCS, vol. 1520, pp. 417–431. Springer Berlin Heidelberg (1998)
8. Cplex Solver: IBM, ILOG, http://www.ilog.com/products/cplex/