Informed Simulated Annealing for Optimizing Dorm Room Assignments

Nguyen Thanh Trung, Tran Ngoc Tuan, Duong Tuan Anh
Faculty of Computer Science & Engineering,Ho Chi Minh City University of Technology
dtanh@cse.hcmut.edu.vn

Abstract

Assigning dorm rooms to 2500 students with complex and interdependent preferences is a difficult optimization problem. In this paper, we present an optimization method for the whole dorm room assignment problem. The novelty of our method is that we apply an improved variant of simulated annealing, called Informed Simulated Annealing (ISA) proposed by Y. Li. We perform experiments of the ISA algorithm on real data sets. The results obtained from the experiments show that the ISA algorithm is more suitable for the dorm room assignment problem.

1. Introduction

The task of assigning students to dorm rooms, taking into account their preferences is a formidable task by hand and is not trivial to automate. For example, at the new dormitory of Ho Chi Minh City (HCMC) University of Technology, every semester, appropriately 2500 students are assigned to 2500 beds in such a way that all student preferences and management policy constraints should be satisfied as much as possible.

Simulated annealing (SA) is one of the well-known meta-heuristics for solving complex constraint satisfaction optimization problems ([4]). Simulated annealing is a variant of local search algorithm which allows non-improving moves to be accepted in a controlled manner. It has been applied successfully in solving several real world applications, including university timetabling ([2], [3]), sport timetabling ([1]), fleet scheduling ([11]), quadratic assignment ([8]), and job shop scheduling ([6]).

There have been several approaches to improve the efficiency of simulated annealing. Some of them are listed as follows. Elmohamed et al. investigated simulated annealing with different cooling schedules ([2]). Poupaert et al. proposed a new simulated annealing variant using acceptance probability as control parameter of the search ([9]). Miki et al. proposed a simulated annealing variant with adaptive neighborhood ([7]). Li developed an enhanced variant of SA, called Informed Simulated Annealing (ISA) which uses the information collected during the annealing search to direct the search ([6]).

There has been one research work that applied SA to dorm room assignment problem (DRAP): the work done by E. Settanni ([10]) for the dormitory at the University of New Mexico. In the project, the author used only standard simulated annealing with some traditional cooling schedules for solving DRAP.

In this work, we explore the new method to solve the DRAP: using ISA, an improved variant of SA. We perform experiments of the algorithm on real data sets from the dormitory at HCMC University of Technology. The experimental results show that the ISA algorithm is quite promising: it brings out a better solution in a shorter time than solving the problem with a standard SA algorithm.

The rest of the paper is organized as follows. In section 2 we describe the dorm room assignment problem. Section 3 presents the main concepts of ISA algorithm. Section 4 describes some related issues such as cooling schedule and neighborhood structure. In Section 5 we present the ISA application to the DRAP, and experimental results obtained. Some conclusions are given in Section 6.

2. The Dorm Room Assignment Problem

The choices that describe the DRAP are a combination of student requests and management regulations. Students specify their preferences for halls, room types, non-smoking/ smoking, music tastes, study habits, and specific requests for roommates on their application forms. The dorm room assignment system is responsible for assigning students to their beds each semester in such a way that all student preferences and management regulations should be satisfied as much as possible. In the DRAP, there are two kinds of

978-0-7695-3580-7/09 $25.00 © 2009 IEEE
DOI 10.1109/ACIIDS.2009.61

Authorized licensed use limited to: University of Johannesburg. Downloaded on June 01,2010 at 19:51:47 UTC from IEEE Xplore. Restrictions apply.

constraints: *hard constraints* that must be satisfied, and *soft constraints* that should be satisfied to improve the solution.

2.1 The Set of Constraints

Hard constraints. At the dormitory of HCMC University of Technology, there are four constraints of this kind.

HC1. *Gender restriction*. The dorm is divided into two quarters: one quarter for male students and the other for female students. So each student must be assigned to a room designated properly for his or her gender.

HC2. *Room capacity restriction*. Depending on room type, a room can accommodate only a limited number of occupants.

HC3. *Privileged students*. The students who are disabled soldiers, or children of dead war heroes are privileged students. The requests from these students must be satisfied.

HC4. *Room change request*. Senior students who are living in the dorm may request to change to better rooms. The requests from these students must be satisfied.

Soft constraints. There are two main kinds of soft constraints: room-student (RS) constraints and student-student (SS) constraints.

i) Room-student constraints take student and room attributes into accounts. For example, a room-student mismatch occurs when a student's hall does not match the hall of his or her choice.

ii) Student-student constraints are concerned only with the students in the room. Student-student errors occur when the occupants of the room do not agree, for example, a student who prefers traditional music is in the same room as someone who objects to this kind of music.

*Room-student Constraints***.** When a student's preference about room is not satisfied, an error occurs. Errors of this kind consist of:

- Unsatisfied preference about hall: w*ERR_HALL
- Unsatisfied preference about room type: w*ERR_ROOM
- Unsatisfied preference about beds: w*ERR_BED
- Unsatisfied preference about rent price: w*ERR_PRICE
- Unsatisfied preference about floor: w*ERR_FLOOR

where *w* is the priority rate for senior students and students with early reservation. *w* is calculated as follows:

senior*SENIORITY_WEIGHT + k

(*senior* is the number of semesters residing in the dormitory and *k* is priority level for early reservations, in the range from 0 to 10).

- Unsatisfied reassignment request: This error occurs if a senior student that wants to remain in his or her room has been assigned to another room. This error is computed as follows:

  ncs*ERR_RE_ASSIGNED

  where *ncs* is the number of continuing semesters residing at the same room.
- Unsatisfied room change request: This error occurs if a senior student that wants to change his or her room has been assigned to the same room. This error is computed as follows:

  # of students * ERR_SAME_ROOM
- Partial fill: This error occurs if any room is not full assigned. This error is computed as follows:

  # of empty beds * ERR_PARTIAL_FILL.

Room-student constraints are divided into two subgroups: subgroup RS1 that consists of constraints regarding to management regulations and subgroup RS2 that consists of constraints regarding to room-attribute preferences.

Student-student constraints. When a student preference about roommates is not satisfied, an error occurs. Errors of this kind consist of:

- Unsatisfied music preference: This error occurs if one student objects to the music that is preferred by another roommate. This error is computed as follows:

  min(# of students who like, # of students who dislike) * ERR_MUSIC
- Unsatisfied study habit: This error occurs if a late studier mixes with non-late studiers in a room. This error is computed as follows:

  min(# of late studiers, # of non-late studiers) * ERR_STUDY_LATE
- Unsatisfied preference about computer game: This error occurs if a student that dislikes playing computer game has to mix with a student that likes playing it. This error is computed as follows:

  min(# of students who like, # of students who dislike) * ERR_PLAY_GAME.
- Unsatisfied smoking preference: This error occurs if a student dislikes smoking has to mix with smoking students in a room. This error is computed as follows:

  min(# of non smoking students, # of smoking students) * ERR_SMOKING.
- Unsatisfied friend preference: This error occurs if the friend one student registers to be his or her roommate is not assigned to share the same room. This error is computed as follows: w*ERR_FRIEND*# of un-satisfied students.

Authorized licensed use limited to: University of Johannesburg. Downloaded on June 01,2010 at 19:51:47 UTC from IEEE Xplore. Restrictions apply.

- Unsatisfied preference about same department roommates: This error occurs if one student's request about roommates from the same department is not satisfied. This error is computed as follows:
  w* # of students of different departments * ERR_DIFF_DEPART
- Unsatisfied preference about same hometown roommates: This error occurs if one student's request about roommates from the same native province is not satisfied. This error is computed as follows:
  w* # of students of different home towns * ERR_DIFF_HOMETOWN
- Unsatisfied preference about same age roommates: This error occurs if one student's request about roommates of the same age is not satisfied. This error is computed as follows:
  w* # of students of different ages * ERR_DIFF_AGE
- Unsatisfied preference about same batch roommates: This error occurs if one student's request about roommates of the same batch is not satisfied. This error is computed as follows:
  w* # of students of different batches * ERR_DIFF_BATCH

Student-student constraints are divided into two subgroups: subgroup SS1 which consists of constraints regarding to personal preferences and subgroup SS2 which consists of constraints regarding to room-mate preferences.

Notice that we have to assign the scores for ERR_HALL, ERR_ROOM, ERR_PRICE, ERR_BED, ERR_MUSIC, ERR_SMOKING, etc. These details are reported in [5].

2.2 Objective Function

The DARP consists in minimizing the objective function which iterates over all the students and sums up their "unhappiness". The objective function f is computed as follows:

$$f(e) = \sum_{s=1}^{m} \Big(\sum_{j\in RS1} \text{RSerr}_j(e,s) + \sum_{j\in RS2} \text{RSerr}_j(e,s) \Big) + \sum_{r=1}^{n} \sum_{s=1}^{t} \Big(\sum_{j\in SS1} \text{SSerr}_j(e,s,r) + \sum_{j\in SS2} \text{SSerr}_j(e,s,r) \Big)$$

where e is the current state of dorm room assignments, m is the number of students, n is the number of rooms, t is the number of students in a room, $RSerr$ is room-student error, $RS1$is the subgroup of room-student constraints related to management regulations, $RS2$ is the subgroup of room-student constraints related to room-attribute preferences, SS_1 is the subgroup of student-student constraints related to student personal preferences, SS_2 is the subgroup of student-student constraints regarding to room-mate preferences, $SSerr$ is a student-student error.

Since the objective function is the sum of errors in a state of dorm room assignments, it is also referred as a *cost function* which can be used to measure how distant a solution is from the optimal solution.

3. Concepts of ISA

Informed Simulated Annealing (ISA), proposed by Y. Li, is an improved variant of SA. ISA is based the relationship between utilities of variable-value pairs in optimal solution and the annealing search that converges to the optimal solution. In this section, we roughly outline the main concepts of ISA. For more details, readers can refer to [6].

Any constraint satisfaction optimization problem (CSOP) consists of a number of variables and a number of associated values for each variable. A *variable-value pair* represents the assignment of a value to a variable. Suppose there are m variables and each variable has n values. Then v_{ij} is the variable-value pair consisting of the jth value in the domain of the ith variable (where $0 < i \leq m$ and $0 < j \leq n$). A solution of a CSOP is a set of variable-value pairs which satisfy all the hard constraints and as many soft constraints as possible.

For SA search, the *repair procedure* is as follows: choose a variable randomly, replace the current value assigned to this variable by randomly choosing a new value in order to remove conflicts. At the beginning of the search, a feasible assignment of values to their variables is randomly generated. This is repaired to a new solution which, if it is accepted, becomes the new current solution for the next repair. If it is not, the old solution remains the current solution and is repaired again. Progressively, solutions move closer and closer to the optimal solution; that is, the search asymptotically converges to the optimal solution.

3.1. Utilities of Variable-Value Pairs

The *utility* of a variable-value pair is an estimation of the probability that the variable-value pair appears in an optimal solution. A straightforward definition of utilities can be based on the *frequency* with which a variable-value pair appears in new current solutions, i.e. the solutions accepted after repairs. The more frequently a variable-value pair appears in new current solution, the higher the probability of its being in the optimal solution.

Let m be the number of repairs in which a variable-value pair v_{ij} is involved. It can be involved in two

Authorized licensed use limited to: University of Johannesburg. Downloaded on June 01,2010 at 19:51:47 UTC from IEEE Xplore. Restrictions apply.

ways, either as the value being repaired or through the value being substituted. Now let *n* be the number of times that v_{ij} appears in the new current solution. Then the desired probability is *n/m*.

Whether a variable-value pair appears in the new current solutions is itself determined by other probabilities. Suppose v_{ij} and v_{ik} are involved in a repair; and suppose *j* is the value being repaired and *k* the value being substituted. Let denote the two solutions as $S(v_{ij})$ and $S(v_{ik})$. Whether $S(v_{ik})$ is accepted is a function of two probabilities. The first probability is calculated by the following function.

$$P1 = \exp(-(f(S(v_{ik})-f(S(v_{ij})))^{+}/T)$$

where $X^{+} = X$ if $X > 0$, and $X^{+} = 0$ otherwise. *P1* specifies the 'general' probability of $S(v_{ik})$'s being accepted as the new current solution. The second probability *P2* is randomly generated and affects whether is actually accepted. It is accepted if and only if $P2 < P1$.

P1 can be used in two ways. First, consider the pair v_{ik}, where *k* is the value substituted for *j* as the assignment to V_i. The probability $P(v_{ik})$ of v_{ik} being in the new current solution is *P1*. This is the probability of accepting $S(v_{ik})$. Second, consider the pair v_{ij}, where *j* is the value repaired. The probability $P(v_{ij})$ of v_{ij} being in the new current solution as $1 - P1$. This corresponds to the probability of rejecting $S(v_{ik})$ as the new current solution.

3.2. Utilities in Practical Annealing Search

Significant Utilities

A utility which is established in practical annealing search is *significant* if it can be used to improve on the best solution found by standard SA. Given an ordering of variable-value pairs for each variable, from the higher to the lower, the task is to find a *threshold* which divides the significant utilities for each variable from the non-significant ones. Note that the variable-value pairs distinguished as significant by their relative utilities will not be assessed individually. Rather they are assessed as *a set*. If the set of pairs serves to lead annealing search to better solution than those found by standard SA, then the utilities of these pairs will be seen as significant.

Directing Search

The set of variable-value pairs distinguished as significant by a *threshold* serves to point to solution better than those found in standard SA search. The basic strategy is to *bias* the search in favor of the set of significant pairs. This is achieved through the neighborhood operator, which *repairs* solutions. At a certain point in the search, the choice of value used to repair the assignment of a variable is biased towards those which appear in the significant pairs for that variable. There is a higher probability of using one of these values than of using any of the non-significant values.

There can be a problem if search is biased towards a few variables or a few variable-value pairs. The basic idea to mitigate this problem is to monitor dynamically the choice of variables and their re-assignments during search. There are two mechanisms that help to achieve this idea. Due to limited space, this paper will not cover it. Interested readers can refer to [6] for more details.

3.3 Implementing ISA

The procedure is that we select a specific threshold, represented by a positive integer *n*, which is to be used for assessing the utilities of variable-value pairs. For each variable we take the top *n* utilities of the variable-value pairs for that variable. These are taken, by definition, to be the significant utilities. These probabilities can vary considerably from variable to variable. The top *n* utilities for each variable pick out a set of variable-value pairs, which is called the *bias set*.

In directed search there are two sets of variable-value pairs which are relevant to each variable, (1) those which belong to the bias set for that variable, and (2) the set of all possible pairs for that variable. Both are used in the repair process of ISA. In the basic repair procedure, there are two basic steps: (1) randomly choose a variable and randomly select a value to replace its current assignment; (2) maintain feasibility by repairing those other variables whose current assignments violate constraints. In ISA step (1) will be modified. At a certain point in the schedule, *ISA will bias the selection of repair values to those which occur in pairs belonging to the bias set.*

To constrain the degree of bias, ISA uses two new probabilities: the first, referred to as P_1, fixes *the degree to which search is biased towards the bias set*; the value of P_1 is fixed from the beginning. The second, referred to as P_2, is used to decide the choice set on a given occasion, i.e., whether it is the bias set or the set of all possible values; this probability is randomly generated during repair. The procedure for applying these probabilities is as follows. At each repair the algorithm randomly chooses a variable, say V_i, and then generates a value for P_2; if it less than P_1, the new variable-value pair v_{ij} must be selected from the bias set, otherwise v_{ij} is selected from the set of all possible values of variable V_i.

Repairing the first variable may cause other variable assignments to violate constraints. These must be repaired to maintain feasibility. Here, we shall not use the bias set in repairing these variables; that is, this will

Authorized licensed use limited to: University of Johannesburg. Downloaded on June 01,2010 at 19:51:47 UTC from IEEE Xplore. Restrictions apply.

be no bias except in the repair of the *initially selected variable*. For all other variables which require repair to maintain feasibility, the values will be selected randomly from the set of all possible values.

One key issue in directing search concerns where the search becomes directed. There are two alternatives: (1) The utilities should be assessed at the end of the basic schedule and (2) A suitable bias set may emerge earlier than one might expect. Y. Li ([6]) proposed two slightly different implementations of ISA.

1. The first, called ISA-1, uses the whole of the basic annealing search to build up the utilities of the variable-value pairs. It then assesses these utilities and initiates further search using a new schedule. This search is directed towards the bias set.
2. The second, called ISA-2, assesses the utilities of pairs before the basic annealing schedule is finished. The bias set is determined at this point and subsequent search is biased towards this set throughout the remainder of the basic schedule.

4. Some Other Issues

To make the DRAP to be readily structured for ISA algorithm, we have to handle some related issues.

4.1Cooling Schedule

In ISA, we use the most common temperature decrement rule:

$$T_{k+1} = \alpha\, T_k$$

Here α is determined by *N*, the predefined number of iterations the user wants for the annealing algorithm. The *reduction parameter* α is derived from *N* using the following formula:

$$T_f = T_0 * \alpha^N$$

where T_0 is the initial temperature and T_f is the final temperature.

Through some mathematical manipulations, we can obtain:

$$\alpha = 1 - (\ln(T_0) - \ln(T_f))/N$$

Another decision we have to make is how many iterations ISA makes at each temperature. A constant number of iterations at each temperature (*nrep*) is a common scheme.

4.2 Neighborhood Operator

From a current state of room assignment s_0, a student is selected randomly, then another student of the same gender is also selected randomly, and their beds are swapped. By that move, we obtain a neighbor state *s* for the current state s_0.

4.3 Two-Stage Solving Method

The approach we take to the DRAP at the dormitory of HCMC University of Technology consists of two stages:

1. Phase I: to obtain an initial state of dorm room assignment which satisfies all the hard constraints.
2. Phase II: to improve the quality of the initial dorm room assignment state, taking the soft constraints into account. The method used in the second phase is optimization method which will seek to optimize the given objective function. We used ISA with the option ISA-1 in this optimization phase.

5. Experimental Results

We implemented the dorm room assignment system with Microsoft Visual C# and experimented on a Pentium IV 2600 Hertz PC with 512 MB RAM. The real data set is from the dormitory at the HCMC University of Technology. The data set consists of 2500 students and about 2500 beds. For each student, there are 3 hard constraints and 19 soft constraints gathered from student dorm registration forms. So the system has in total about 9000 hard constraints and 57000 soft constraints to be considered. The database which keeps information about students, rooms, beds, student-preferences and dorm room assignments is implemented on SQL Server 2000.

After some experiments with different parameter values, we found out that the system brings out good quality solution with the parameters as follows: T_0 = 10000, T_f = 0.0001 and *nrep* = 6. The run times for different numbers of iterations through the tests on real data set are given in Table 1. According to the experiments, the system can yield good quality solution after 10000 iterations and in about 2 hours. This time complexity is acceptable since the system is used to generate dorm room assignments every semester.

Table 1. The run times of ISA for some different numbers of iterations

Number of iterations	500	1000	5000	10000	20000
Run time (in minutes)	5	12	56	122	268

The anytime cure which illustrates the change of objective function value (also called the solution cost) along the number of iterations is given in Figure 1. Since the DRAP is an over-constrained problem, during the execution of ISA we can gradually reduce the cost but we can not make it equal to 0.

Compared with the previous approach which uses standard simulated annealing and tests on the same real

Authorized licensed use limited to: University of Johannesburg. Downloaded on June 01,2010 at 19:51:47 UTC from IEEE Xplore. Restrictions apply.

data sets ([5]), the ISA take a shorter time and yields a quite better solution. In Table 2, the running times (in minutes) of the two methods (ISA and standard SA) can be compared. We can see that the application of ISA to the dorm room assignment results in a two times speedup over standard SA.

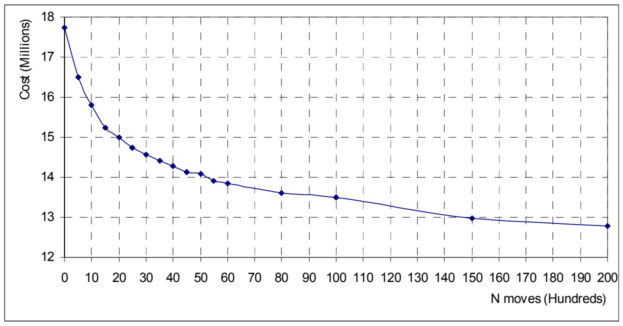

Figure 1. Effect of the number of iterations on objective function value.

Table 2 ISA versus standard SA for DRA problem

Number of iterations	500	1000	5000	10000	20000
ISA	5	12	56	122	268
Standard SA	15	27	130	245	470

Since the DRAP varies from one institution to the other, it is difficult to compare our results with that of the previous work in the world. However, we believe that ISA performs better than standard SA in this particular problem.

6. Conclusions

ISA is an improved variant of SA which employs learning mechanism to gather information and uses the information to direct the search. We successfully applied ISA to the dorm room assignment problem at the dormitory of HCMC University of Technology. Experiments carried out on real data sets demonstrated that ISA increases the speed at which better solutions are found above standard simulated annealing.

Our main conclusion from this work is that dorm room assignment problems represents a class of problems in which repair is "local" process and ISA performs much better than standard SA in this type of problems. In other words, ISA is an appropriate approach to the dorm room assignment problem, which is a very hard combinatorial optimization problem.

In a future work we plan to explore some newly proposed improved variants of simulated annealing to the dorm room assignment problem in order to find out which variant of SA is the best performer for this particular problem.

Acknowlegments

This research has been financially supported by Vietnam National University of Ho Chi Minh City.

REFERENCES

[1] A. Anagnostopoulos, L. Michel, P. Van Hentenryck, and Y. Vergados, "A Simulated Annealing Approach to the Traveling Tournament Problem", In: *Proceedings of CPAIOR'03*, 2003.

[2] M.A. S. Elmohamed, G. Fox, P. Coddington, "A Comparison of Annealing Techniques for Academic Course Scheduling", In: *Proc of 2nd International Conference on the Practice and Theory of Automated Timetabling (PATAT'97)*, 1997,146-166.

[3] T. A. Duong, K. H. Lam, "Combining Constraint Programming and Simulated Annealing on Exam University Timetabling", In: *Proc. of 2nd International Conference RIVF'04 Research Informatics Vietnam-Francophony*, Hanoi, Vietnam, Feb. 2004, 205-210.

[4] F. Glover & G.A. Kochenberger, "*Handbook of Metaheuristics*", Kluwer Academic Publishers, 2003.

[5] L. T. Hieu, "Simulated Annealing Approach for Dorm Room Assignment Problem", Master Thesis, Dept. of Information Technology, HoChiMinh City University of Technology, Dec. 2004.

[6] Y. Li, "Directed Simulated Annealing In Constraints Satisfaction And Optimization", Ph.D. Thesis, Imperial College of Science, Technology and Medicine, University of London, October 1997.

[7] M. Miki, T. Hiroyasu, K. Ono, "Simulated Annealing with Advanced Adaptive Neighborhood", In: *Proc. of 2nd Workshop on Intelligent Systems Design and Application*, 2002, Atlanta, Georgia.

[8] A. Misevicius, "A Modified Simulated Annealing Algorithm for Quadratic Assignment Annealing", *Informatica*, vol. 14, No. 4, 2003, 497-514.

[9] E. Poupaert, Y. Deville, "Simulated Annealing with Estimated Temperature", *AI Communications*, Vol 13, 2000, 19-26.

[10] E. Settanni, "Improving Dorm Room Assignments Using Simulated Annealing", Master thesis, Dept. of Computer Science, The University of New Mexico, Albuquerque, New Mexico, Dec. 2000.

[11] D. Sosnowska, and J. Rolim, "Fleet Scheduling Optimization: A Simulated Annealing Approach", In: *Proc. of 3rd Int. Conf. on the Practice and Theory of Automated Timetabling (PATAT'2000)*, Konstanz, Germany, August 2000, (Eds. E. Burke and W. Erben), LNCS 2079, Springer-Verlag, 2001, 277-294.

Authorized licensed use limited to: University of Johannesburg. Downloaded on June 01,2010 at 19:51:47 UTC from IEEE Xplore. Restrictions apply.