Minimal Perturbation Problem – A Formal View[*]

Roman Barták[1], Tomáš Müller[1], Hana Rudová[2]

[1] Charles University, Faculty of Mathematics and Physics
Malostranské nám. 2/25, Prague, Czech Republic
{bartak, muller}@ktiml.mff.cuni.cz

[2] Masaryk University, Faculty of Informatics
Botanická 68a, Brno, Czech Republic
hanka@fi.muni.cz

Abstract. Formulation of many real-life problems evolves as the problem is being solved. These changes are typically initiated by a user intervention or by changes in the environment. However, the traditional formulation of the constraint satisfaction problem is static in the sense that it must be fully specified before the solving process starts. In this paper, we propose a formal description of so called minimal perturbation problem that allows an "automated" modification of the (partial) solution when the problem formulation changes. We also discuss a first proposal of the algorithm for solving such type of problems.

Introduction

Many real-life problems can be naturally stated as constraint satisfaction problems (CSP). To further spread up applicability of the constraint satisfaction technology in real-life applications, it is necessary to cover dynamic of the real world. It means that quite often the problem formulation is not static but it evolves in time. In fact, it evolves even during solving the problem. The changes in the problem formulation can be initiated by the user who reacts to the (partial) solution found so far by proposing some changes. Or the problem changes result from the changes in the environment like broken machines, delayed flights etc. Naturally, the problem solving process should continue as smoothly as possible after any change. In particular, the solution of the altered problem should not differ too much from the (partial) solution found for the initial problem. There are several reasons to keep a new solution as close as possible to the initial solution. For example, if the initial solution has already been published (like the assignment of gates to flights) then it would not be nice to change it frequently from the obvious reasons. Moreover, changes to the published solution may tend to push to other changes and so on. The paper studies this new type of the problem called a *minimal perturbation problem* (MPP), i.e., finding a solution of the

[*] Research is supported by the Grant Agency of the Czech Republic under the contract no. 201/99/0942 and by Purdue University.

new problem in such a way that this new solution does not differ much from the solution of the initial problem.

Our work is motivated by a large scale timetabling problem at Purdue University, USA. The primary focus of this work is to provide a support for making changes to the generated timetable. Once timetables are published they require many changes based on additional user input. These changes should be incorporated into the problem solution with a minimal impact on any previously generated solution. We are already able to generate the solution for the initial problem [8] and our current work consists of solving the new problem after the user interaction. The basic requirement is to keep the solution as close as possible to the published solution of the initial problem, of course provided that the new solution is a solution of the altered problem. Formally it means that the minimal number of variable assignments is changed after the problem modification.

The solver for the initial problem is based on the limited assignment number (LAN) search algorithm [10]. This algorithm tries to assign a maximal number of variables in such a way that the resulting problem is still consistent. The algorithm does not guarantee finding a complete solution but it provides a good partial solution in a limited time. We intend to extend this (primary) search algorithm in the direction towards MPP using hybridization with other (secondary) search algorithm. We will discuss the main ideas behind such extension and we will describe the desired properties of the other (secondary) algorithm.

The paper is organized as follows. We first give more details about the lecture timetabling problem that makes the main motivation of our research. Then we survey the existing approaches to the minimal perturbation problem. The main part of the paper is dedicated to a general formalization of the minimal perturbation problem and to ideas behind the solving algorithm for such a problem.

Motivation: A Course Timetabling Problem

The primary intent behind our work on minimal perturbation problems lies in the need to solve such a problem in the context of a real timetabling application, in particular a timetabling solver for Purdue University (USA). Purdue University timetabling problem, that we should solve, consists of timetabling approximately 750 classes into 41 large lecture rooms with capacities up to 474 students. The classes are taught several times a week resulting in about 1,600 meetings to be timetabled. The space covered by all meetings fills approximately 85% of the total available space. Special meeting patterns defined for each class restrict possible time and location placement, e.g., all meetings of the same class must be taught at the same classroom and hour, valid combinations of days are given, etc. Classroom allocation must respect instructional requirements and preferences of faculty. All instructors may have specific time requirements and preferences for each class. A major objective is to minimize the number of potential student course conflicts for each of almost 29,000 students – student course selections are utilized to construct a timetable that attempts to maximize the number of satisfied course requests. However, the main goal is to timetable all classes since there are too many hard constraints.

The construction of the solution for this problem was described in [8]. The main ideas behind the problem solver are as follows. Each meeting of a class is described using two domain variables: a time variable (starting time of the meeting in a week) and a classroom variable. The hard constraints ensure that two meetings will not be taught in the same classroom at once. The constraints also remove values prohibited by meeting patterns and by the requirements of instructors and faculties. Preferential requirements on time and classroom placement together with the student course conflicts are expressed using soft constraints. The cost of a soft constraint ensuring that two classes with common students will not overlap is equal to the number of common students. The preferences of instructors and faculties are expressed using soft unary constraints; the preference of a certain value is expressed as a cost of the constraint demanding a particular value to be chosen.

Labeling is done via a new limited assignment number search (LAN) algorithm [10]. LAN Search is an incomplete iterative search algorithm where the standard backtracking is limited to an incomplete search of the linear complexity. The aim is to generate a partial solution with the maximal number of assigned variables. The algorithm runs in iterations where each iteration step explores some sub-tree of the search tree. The linear complexity is achieved by considering a limit on the number of assignments for each variable tried during the iteration step. The result of each iteration step is a partial solution that is used as a guide in the next iterations. Special value and variable ordering heuristics have been proposed for this purpose.

It may still happen that the set of hard constraints is over-constrained. The user input can be used to resolve this problem by relaxing some constraints. Then the LAN search algorithm can continue in the subsequent iterations with the problem definition changed. This approach is similar to the problem considered in this paper – we have a partial solution and we want to construct a new solution under the redefinition of the problem. However, the approach of LAN search does not minimize the number of changes in subsequent solutions but it maximizes the number of labeled variables. A solver for MPP should do both tasks.

The solution of the timetabling problem was implemented in SICStus Prolog CLP(*FD*) library [1] with the help of our soft constraints solver [9]. The initial implementation started with built-in backtracking of SICStus Prolog. However, the experiments with standard backtracking did not lead to a solution after 10 hours of run time (too many failed computations were repeated exploring the parts of the search tree with no solution). LAN search was able to substantially improve on the initial partial solution. Starting from 33 classes, only one class remained unassigned after eight iterations of LAN search. Assignment of this class was successfully completed with the help of the user.

The solution generated by the LAN Search algorithm introduces the initial solution. Once timetables are published they require many changes based on additional input. These changes should be incorporated into the problem solution with a minimal impact on any previously generated solution. Note, that it is not possible to limit the changes in the problem definition. Faculties and instructors may come with completely new classes to be scheduled and the requirements towards the original classes may change substantially. Also some requirements or classes can be canceled which can help to find a more acceptable solution. On the other hand, the new requirements usually make the problem harder since we are constrained in the number of allowed changes with respect to the original solution. The original problem, its solution, and the set of desired changes introduce the input for the minimal perturbation problem we are going to solve.

LAN search algorithm introduces the very first step in the direction towards solving the minimal perturbation problem. The partial solution generated in each iteration step is derived from the former solution. As our experiments showed, the distance of these solutions is mostly acceptable when a small number of changes is done in the problem formulation. However, the distance significantly enlarges if the number of desired changes increases. Other experiments also showed that the more informed ordering heuristics may improve the quality of the generated solution substantially. Thus LAN search algorithm seems to be a good base for solving of minimal perturbation problem.

Related Works

The minimal perturbation problem (MPP) is not a new notion. This type of problems appears frequently in real-life planning and scheduling applications where the task is to “minimally reconfigure schedules in response to a changing environment” [3]. The *minimal perturbation problem* was described formally by El Sakkout, Richards, and Wallace in [2] as a 5-tuple $\Pi = (\Theta, \alpha, C_{add}, C_{del}, \delta)$ where:

- Θ is a CSP (i.e. a triple (V,D,C), where V is a set of variables, D are domains for V, and C is a set of constraints);
- α is a solution to Θ (i.e., a complete assignment satisfying the constraints from C)
- C_{add}, C_{del} are constraint removal and addition sets;
- δ is a function that measures the distance between two complete assignments (perturbation).

A complete assignment β is a solution to Π iff it is a solution to CSP (V, D, C^*), where $C^* = (C \setminus C_{del}) \cup C_{add})$, and $\delta(\alpha,\beta)$ is minimal.

Notice that the above formulation of MPP is for hard CSPs where all the constraints must be satisfied by a complete assignment of variables. Moreover, it allows addition and retraction of constraints only so the set of variables is not changing.

Our view of MPP differs from the above definition in several ways. First, we formulate MPP for soft CSPs, i.e., the best incomplete assignments are compared. Second, we allow more general changes in the problem formulation; in particular both the set of constraints and the set of variables (together with domains) can be changed. Last but not least, our definition of the function δ measuring the distance between the assignments is more concrete in comparing differences in the assignments.

A Formal Model

In this section, we present a new formal model of the minimal perturbation problem that is applicable to over-constrained problems as well as to problems where finding a complete solution is hard. Recall that the idea of MPP is to define a solution of the altered problem in such a way that this solution is as close as possible to the (partial) solution of the original problem. We first survey the standard definitions of CSP and we introduce a new notion of (locally) maximal consistent assignment. In the second part, we formally define a minimal perturbation problem and a solution of this problem.

Preliminaries

A *constraint satisfaction problem (CSP)* is a triple Θ=(V,D,C), where

- V= {v_1,v_2,…,v_n}is a finite set of variables,
- D={D_1,D_2,…,D_n}}is a set of domains (i.e., D_i is a set of possible values for the variable v_i),
- C= {c_1,c_2,…,c_m} is a finite set of constraints restricting the values that the variables can simultaneously take.

A *solution* to the constraint satisfaction problem Θ is a complete assignment of the variables from V that satisfies all the constraints.

For many problems it is hard or even impossible to find such a solution. For example, for over-constrained problems [4], there does not exist any complete assignment satisfying all the constraints. Therefore other definitions of problem solution like Partial Constraint Satisfaction were introduced [4]. We introduce here a new notion of maximal consistent assignment that is motivated by the university timetabling problem but we believe that it has a more general usage. The basic idea behind is to assign as many as possible variables while still keeping the rest of the problem “consistent”. It means that the user may relax some constraints in the problem (typically some of the constraints among the non-assigned variables that cause conflicts) so that after this change the assignment can be extended to other variables.

Formally, let Θ be a CSP and C be a consistency technique (for example arc consistency). We say that the *constraint satisfaction problem* is *consistent* if the consistency technique deduces no conflict (e.g., for arc consistency, the conflict is indicated by emptying some domain). We denote C(Θ) the result of the consistency test which could be either true, if the problem Θ is C consistent, or false otherwise. Let Θ be a CSP and σ be a (partial) assignment of variables, then we denote $\Theta\sigma$ application of the assignment σ to the problem Θ, i.e., the domains of the variables in σ are reduced to a singleton value defined by the assignment. Finally, we say that a *partial assignment s* is *consistent* with respect to some consistency technique C iff C($\Theta\sigma$). Note that a complete consistent assignment is a solution of the problem. Note also that the backtracking-based techniques typically extend a partial consistent assignment towards a complete (consistent) assignment.

As we already mentioned, for some problems there does not exist any complete consistent assignment; these problems are called over-constrained. In such a case, we propose to look for the maximal consistent assignment. We say that the *consistent assignment is maximal* for given CSP if there is no other consistent assignment with a larger number of assigned variables. We can also define a weaker version, so called *locally maximal consistent assignment.* Locally maximal consistent assignment is a consistent assignment that cannot be extended to another variable(s). Notice the difference between the above two notions. The maximal consistent assignment is defined using the cardinality of the assignment (the number of assigned variables) so it has a global meaning while the locally maximal consistent assignment is defined using a subset relation, i.e., it is not possible to assign an additional variable without getting inconsistency. It is pretty easy (fast) to extend any consistent assignment to a locally maximal consistent assignment. In fact, every branch of the search tree defines such a locally maximal consistent assignment. Visibly, the maximal consistent assignment is the largest (using cardinality) locally maximal consistent assignment.

If the constraint satisfaction problem has a solution then any maximal consistent assignment is the solution. Thus, looking for a maximal consistent assignment is a general way of solving CSPs because it covers both standard CSPs as well as over-constrained problems. Moreover, it is not necessary to know in advance whether the problem is over-constrained or not. Still, for some problems it may be hard to find a maximal consistent assignment. In such a case, we propose to return the largest locally maximal consistent assignment that can be found using given resources (e.g., time). This approach has a strong real-life motivation, for example for timetabling and scheduling problems [6,8] it means that the system allocates as many activities as possible in given time (and no more activity can be allocated without a change of the current allocation). Typically, the solving algorithms based on the above idea select some sub-space of the solution space and for this sub-space they find a maximal consistent assignment which is a locally maximal consistent assignment in the original solution space. For example, the LAN Search algorithm [10] restricts the number of assignments tried per variable.

A Minimal Perturbation Problem

Now we can formally define a *minimal perturbation problem* (MPP) as a quadruple $\Pi = (\Theta, \Theta', F, \alpha)$, where:

- Θ, Θ' are two CSPs,
- F is a mapping of the variables from Θ to Θ' (see below for details), and
- α is a (locally) maximal consistent assignment for Θ called initial assignment.

The function F defines how the problem Θ is changed in terms of the variables. It is (almost) one-to-one mapping of the variables from Θ to the variables from Θ'. For some variables *v* from Θ, the function F might not be defined which means that the variable *v* is removed from the problem. However, if the function F is defined then it is unique (it is a one-to-one mapping), i.e., $v \neq u$ & $!F(v)$ & $!F(u) \Rightarrow F(v) \neq F(u)$. Also, for some variables *v*' from Θ', the origin might not be defined (i.e., there is no variable v such that $F(v) = v'$), which means that the variable v' is added to the problem. Notice also that the constraints and domains can be changed arbitrarily when going from Θ to Θ'. We do not need to capture such changes using the mapping functions like F because we concern primarily about the variable assignments.

Let σ be a (partial) assignment for Θ and γ be a (partial) assignment for Θ'. Then we define $W_\Pi(\sigma,\gamma)$ as a set of variables v from Θ such that the assignment of v in σ is different from $F(v)$ in γ, i.e., $W_\Pi(\sigma,\gamma) = \{v\in\Theta \mid v/h\in\sigma \;\&\; F(v)/h'\in\gamma \;\&\; h\neq h'\}$[1]. We call $W_\Pi(\sigma,\gamma)$ a *distance set* for σ and γ in Π.

A *solution to the minimal perturbation* problem $\Pi = (\Theta, \Theta', F, \alpha)$ is a (locally) maximal consistent assignment β for Θ' such that the size of the distance set $W_\Pi(\alpha,\beta)$ is minimal. The idea behind the solution of MPP is apparent – the task is to find the best possible assignment of the variables for the new problem in such a way that it differs minimally from the existing variable assignment of the initial problem.

Let us summarize now the two criteria used when solving MPP: the first criterion is maximizing the number of assigned variables, the second criterion is minimizing the number of differences between the resultant solution and the previous (initial) solution over the variables which are both in the previous and in the new CSP. So, the candidates for the solution are ordered lexicographically (max, min).

Example:

Let $\alpha=\{a/1,b/3\}$ be an initial solution of a CSP Θ with variables $V=\{a,b,c\}$ and Θ' be a new CSP with variables $V'=\{b,c,d\}$, domains $D' = \{D_b=\{1,3\}, D_c=\{1,2,3\}, D_d=\{2,3\}\}$, and constraints $C'=\{b\neq c, c\neq d, d\neq b\}$. Assume that there is a mapping $F:\{b\to b, c\to c\}$ of variables from Θ to Θ'. Then the problem Θ' has the following solutions (maximal consistent assignments):

- $\beta_1 = \{b/1,c/2,d/3\}$ ($W_\Pi(\alpha,\beta_1) = \{b\}$),
- $\beta_2 = \{b/1,c/3,d/2\}$ ($W_\Pi(\alpha,\beta_2) = \{b\}$),
- $\beta_3 = \{b/3,c/1,d/2\}$ ($W_\Pi(\alpha,\beta_3) = \{\}$),

but only the solution β_3 is the solution of MPP $\Pi = (\Theta, \Theta', F, \alpha)$.

Note finally that we can adapt the ideas for solving CSPs presented in the previous section to solving MPPs. In particular, we can restrict the solution space and then we can look for the maximal consistent assignment within this sub-space. Typically, the restriction of the solution space will be driven by the minimal distance of included (partial) assignments from the initial assignment in Π. We present some ideas of the solving algorithm in the next section.

Ideas of the MPP Solver

As we sketched above, the MPP solver may share some ideas behind the solvers exploring locally maximal consistent assignments. In particular, the MPP solver may look for the maximal consistent assignment in a sub-space of the solution space defined by the distance of assignments from the initial assignment. LAN Search is an

[1] For simplicity reasons we write $v\in\Theta$ which actually means $v\in V$, where $\Theta = (V,D,C)$.

example of the algorithm driven by such principles. Thus our first idea is to extend/refine this algorithm in such a way that the algorithm is able to continue solving the problem after its change. In particular, we intend to insert a *secondary algorithm* that would propose assignments to the primarily backtracking-based search algorithm (we call the values from this assignment *recommended values*). This proposal follows the results from [7] where it has been shown theoretically that backtracking-based search guided by information about a complete assignment has a better average running time. Naturally, the secondary algorithm should propose an assignment close to the initial assignment but respecting in some sense the new problem (the new constraints). Thus we can say that the secondary algorithm defines a value ordering heuristic for the primary algorithm.

We expect the secondary algorithm to run very fast and thus it can be called several times by the primary algorithm (a dynamic value ordering heuristic). Still, the secondary algorithm will not have a trivial (constant) complexity and thus the calls to it should be restricted somehow. Our first intent is to call this algorithm before the primary search algorithm starts and then every time a domain of the currently labeled variable does not contain a recommended value (such value might be removed by some look-ahead propagation technique). In this second case, the secondary algorithm will propose a new value for the currently labeled variable and it will change the recommended values for future (not-yet assigned) variables accordingly.

To achieve fast execution of the secondary algorithm, we expect it to solve a portion of the problem only. In particular, only some (easier) constraints from the problem will be assumed by the secondary algorithm. One of the ideas is to use local search techniques [11] for the secondary algorithm that can propose good assignments fast. This technique has already been applied in [5] where local search was used in each step of the construction of the partial assignment.

Requirements on the Secondary Algorithm

The secondary algorithm will propose an assignment of the variables that will guide the primary algorithm (like a value ordering heuristic). Naturally, this recommended assignment will be probably inconsistent otherwise we have a solution and no primarily algorithm is required. Still the assignment recommended by the secondary algorithm should be “partially” consistent and it should be as close as possible to the initial assignment. To achieve the partial consistency, we simply relax the constraint satisfaction problem by removing some constraints. Then the secondary algorithm should return the maximal consistent assignment for this relaxed problem.

Formally, let $\Pi = (\Theta, \Theta', F, \alpha)$ be a minimal perturbation problem, such that $\Theta' = (V,D,C)$, and β be a partial consistent assignment of variables for Θ' found so far. Then we construct a new MPP $\Pi' = (\Theta, \Theta'', F, \alpha)$ such that $\Theta'' = (V,D,S(C))\beta$, where $S(C)\subseteq C$. Basically we remove “some” constraints from the new problem using a selection function S and we apply the partial assignment β to this relaxed problem. Let $K = |W_\Pi(\alpha,\beta)|$, i.e., K describes the current number of differences between the initial assignment and the new partial assignment. We require the secondary algorithm to find a maximal consistent assignment γ for the problem Θ'' such that $|W_{\Pi'}(\alpha,\gamma)| \leq K+L$ for some given natural number L. It means that the secondary algorithm can add L additional differences to the recommended assignment. In particular if L=0 then the secondary algorithm solves the MPP Π'. However, γ is not necessarily a solution of MPP Π; it just extends β in the most promising way.

Conclusions

The paper presents a new view of the minimal perturbation problem motivated by a real-life application of university course timetabling. This new framework supports incomplete assignments as a problem solution which is useful to model over-constrained problems or hard-to-solve problems. Moreover, this new formulation allows looser changes of the problem formulation like addition/retraction of constraints and variables as well as changes in the variables' domains. We have also sketched the preliminary ideas of the solving algorithm for MPP. Next work will concentrate on the design and implementation of the solving algorithm and on testing the algorithm in a real-life timetabling application.

References

1. Mats Carlsson, Greger Ottosson, and Bjorn Carlson. *An open-ended finite domain constraint solver.* In Programming Languages: Implementations, Logics, and Programming. Springer-Verlag LNCS 1292, 1997.
2. Hani El Sakkout, Thomas Richards, and Mark Wallace. *Minimal Perturbation in Dynamic Scheduling*. In H. Prade (editor): 13th European Conference on Artificial Intelligence. ECAI-98. John Wiley & Sons, 1998.
3. Hani El Sakkout and Mark Wallace. *Probe Backtrack Search for Minimal Perturbation in Dynamic Scheduling*. Constraints 4(5): 359-388. Kluwer Academic Publishers, 2000.
4. Freuder, E.C., Wallace R.J., Partial Constraint Satisfaction, *Artificial Intelligence*, 58:21-70, 1992.
5. Narendra Jussien and Olivier Lhomme. Local search with constraint propagation and conflict-based heuristics. Artificial Intelligence, 139(1):21-45, 2002.
6. Tomáš Müller and Roman Barták. *Interactive Timetabling: Concepts, Techniques, and Practical Results.* In E. Burke, P. De Causmaecker (eds.): Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT2002), Gent, 2002, pp. 58-72.
7. Paul Walton Purdom, Jr. and G. Neil Haven. *Probe order backtracking.* SIAM Journal on Computing, 26(2):456-483, 1997.
8. Hana Rudová and Keith Murray. *University Course Timetabling with Soft Constraints.* Practice And Theory of Automated Timetabling, Selected Papers. Springer-Verlag LNCS, 2003. Accepted for publication.
9. Hana Rudová, *Soft CLP(FD)*. In Susan Haller and Ingrid Russell (eds.): Proceedings of the 16th International Florida Artificial Intelligence Symposium, FLAIRS-03, AAAI Press, 2003. Accepted for publication.
10. Hana Rudová and Kamil Vermirovský, *Limited Assignment Number Search Algorithm.* Submitted to CP'03 conference.
11. Stefan Voß. *Meta-heuristics: State of the art.* In Alexander Nareyek, editor, Local search for planning and scheduling: revisited papers, pp. 1-23. Springer-Verlag LNAI 2148, 2001.