Stable Solutions dealing with Dynamics in Scheduling based on Dynamic Constraint Satisfaction Problems

Hiromitsu Hattori[1], Toramatsu Shintani[1], Atsushi Isomura[1], Takayuki Ito[1], and Tadachika Ozono[1]

Graduate School of Engineering, Nagoya Institute of Technology
Gokiso-cho, Showa-ku, Nagoya, Aichi 466-8555, JAPAN
{hatto,tora,isomura,itota,ozono}@ics.nitech.ac.jp

Abstract. The real-life scheduling problems are often over-constrained, and there is often one intractable case where unexpected events are added and a sudden change is required. In this paper, we describe such problems as the Dynamic Valued Constraint Satisfaction Problem (DyVCSP). In DyVCSP, although the previous schedule would be modified when there is some change, the new schedule should be similar to the current one. We propose a method for achieving solution stability, which maintains portions of the current schedule using the provisional soft constraint. The provisional constraint ensures each variable keeps its value as much as possible. In this paper, we formalize the nurse scheduling problem as DyVCSP and use our method to achieve solution stability.

1 Nurse Scheduling as a Dynamic VCSP

In a DyVCSP, the nurse scheduling problem could be defined as a sequence of VCSPs, each of which represents the problem at each time step. Each VCSP is changed to the next VCSP by loss or gain of values, variables, or constraints. A VCSP at a time step i is defined by $\mathcal{VP}_i = (X_i, D_i, C_i, S, \varphi)$, where $X_i = \{x_{(1,1)}, x_{(1,2)}, ..., x_{(s,t)}, ...\}$. $x_{(s,t)}$ is the working shift of nurse s on day t. D_i is the set of domain. $d_{(s,t)}$ is the domain of $x_{(s,t)}$. On the nurse roster, $d_{(s,t)}$ is $\{free, morning, evening, night\}$. S is the valuation structure defined by $E = [0, 9]$, $\succ = >$, $\perp = 0$, $\top = 9$, $\otimes = +$, and the valuation function φ is simply summation of values. C_i is the set of constraints. The constraints are described in the following form:

$$constraint(lower\_lim, upper\_lim, assignment\_list, weight).$$

This constraint is satisfied if the number of elements in the current assignment corresponding to those in the *assignment_list* are more than the *lower_lim* and less than the *upper_lim*. This is an example of a personal constraint:

$$constraint(1, 3, \{x_{(s,1)} = morning, ..., x_{(s,t)} = morning, ...\}, 4)$$

This is satisfied if the number of corresponding elements between the current assignment and the *assignment_list* is 1 to 3. Then, a DyVCSP could be described as follows:

$$\mathcal{DP} = \{\mathcal{VP}_0, \mathcal{VP}_1, ..., \mathcal{VP}_i, ...\}$$

The problem with solution stability [1, 2] in DyVCSP can be defined as the problem of sequentially computing a solution for each of the VCSPs $\{\mathcal{VP}_0, \mathcal{VP}_1, ..., \mathcal{VP}_i, ...\}$ given some existing static constraint satisfaction algorithms.

2 Re-scheduling with Dynamic VCSP

We focus on a case where a nurse suddenly needs to change his/her schedule and re-scheduling is needed. As mentioned above, we deal with the solution stability. In this paper, the solution stability is defined as follows:

solution stability : The results after re-scheduling should be similar to those of previously completed scheduling because users are confused if there are extensive changes.

In our method for solution stability, we introduce a provisional constraint. The provisional constraint is used to maintain the previous value of each variables as much as possible. For example, when $v_{(i,j)}$ is assigned to the variable $x_{(i,j)}$ in the previous problem, the provisional constraint which is used to keep the value $v_{\,(i,j)}$ is:

$$constraint(1, 1, \{x_{(i,j)=v_{(i,j)}}\}, w)$$

where, w is the weight of the provisional constraint. The value of w is predefined.

The process of re-scheduling for achieving solution stability is as follows:

Step 1: New constraints for sudden requests are generated and the constraints C_{new} are added to current problem $\mathcal{VP}_i$. The problem then changes $\mathcal{VP}_i$ to $\mathcal{VP}_{i+1}$. The new constraints C_{new} is the cause of re-scheduling.

Step 2: The provisional constraints C^i_{prov} to maintain the current solution are generated and added to the set of provisional constraints C_{prov}, which consists of all provisional constraints. C_{prov} is as follows:

$$C_{prov} = \bigcup_{j=0}^{i} C^j_{prov} \ \ (\forall j \ c \in C^j_{prov}, c \notin C_{prov})$$

Step 3: C_{prov} is added to $\mathcal{VP}_{i+1}$. Then, $\mathcal{VP}_{i+1}$ is changed to $\mathcal{VP}'_{i+1}$. If the summation of the weight of newly added provisional constraints $(W_{C^i_{prov}})$ is higher than that of the weight of C_{new} in Step 1 ($W_{C_{new}}$), the re-scheduling process stops. This is because the schedule is not modified when $W_{C_{new}} \leq W_{C^i_{prov}}$.

Step 4: The problem $\mathcal{VP}'_{i+1}$ is solved based on basic stochastic hill climbing. Since the provisional constraints that keep the previous value of each variable are included in Step 3, a stable solution would be obtained.

Step 5: All provisional constraints included in C_{prov} are removed, and then the problem $\mathcal{VP}'_{i+1}$ is changed back into $\mathcal{VP}_{i+1}$. Here, all satisfied provisional constraints are removed from C_{prov} to avoid duplication of them in solving $\mathcal{VP}_{i+2}$.

References

1. Wallace, R.J., Freuder, E.C.: Stable solutions for dynamic constraint satisfaction problems. In: Proceedings of the 4th International Conference on Principles and Practice of Constraint Programming. (1998) 447–461
2. Verfaillie, G., Schiex, T.: Solution reuse in dynamic constraint satisfaction problems. In: Proceedings of the 12th National Conference on Artificial Intelligence(AAAI-94). (1994) 307–312