Solving a Judge Assignment Problem Using Conjunctions of Global Cost Functions

Simon de Givry[1] and J.H.M. Lee[2] and K.L. Leung[2] and Y.W. Shum[2]

[1] INRA, Centre de recherches de Toulouse, Station de biomtrie et d'intelligence artificielle, Chemin de borde rouge, B.P. 52627 - 31326 CASTANET-TOLOSAN CEDEX FRANCE
Simon.Degivry@toulouse.inra.fr

[2] Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
{jlee,klleung,ywshum}@cse.cuhk.edu.hk

Abstract. The Asia Pacific Information and Communication Technology Alliance (APICTA) Awards has been held for 12 years, rewarding the most innovative solutions in different categories of Information and Communication Technology (ICT). To maintain professionalism, judges are nominated from each economy, and appointed to panels of different categories. Judge assignment is a difficult task, since it has to optimize between expertise, distribution of workloads, fairness and sometimes even political correctness. In this paper, we describe our experience in analyzing and automating the APICTA judge assignment process using Soft Constraint Programming for the 13^{th} APICTA hosted in Hong Kong on November, 2013. We chose the *weighted constraint satisfaction* (WCSP) framework since both hard constraints and preferences can be modeled by cost functions. Consistency algorithms can effect strong propagation by redistributing costs among cost functions. We observe that a number of restrictions in the judge assignment problem involves counting. In our first attempt, we utilized the Soft_Among^{var} global cost function for these counting conditions but we could not solve the problem within a day. Soft_GCC^{val} is another possible global cost function to model counting, which is what we used in the second attempt. We can compute the optimum in a few hours, which is far from practical.
We apply similar techniques as Régin to show that the combination of Soft_GCC^{val} and Soft_Among^{var} is flow-based. We further prove that the combination results in a *flow-based projection-safe* cost function, meaning that soft arc consistencies can be enforced efficiently. By using this combination in our final model, we can solve the judge assignment problem within a few minutes. We consider this a success story where theory and practice meet.

1 Introduction

The Asia Pacific Information and Communication Technology Alliance (APICTA) Awards[3] is an international Awards Program. It aims at increasing the awareness of Information and Communication Technology (ICT) in the community,

[3] http://www.apicta.org/

facilitating technology transfers, and offering business matching opportunities, by providing networking and product benchmarking opportunities to ICT innovators and entrepreneurs. Up to 2013, 13 economies have joined the APICTA Awards, including China, Malaysia, Thailand, to name a few. In 2013, the APICTA Awards was hosted in Hong Kong[4]. Preparation becomes a hard work due to limited resources. The APICTA Awards organizers require planning for two different scenarios.

1. Scheduling the presentations for nominated candidates, and;
2. Assigning judges into panels for each award category representing different aspects of the current ICT fields.

The former subjects to venue and time constraints, which has been completed beforehand. The latter involves various logistical and political factors, which are optimization in nature. Judges are nominated by different economies. After nomination, judges of each category are selected by a standard procedure to ensure professionalism and fairness. However, the manual procedure is tedious and inefficient, resulting in large number of complaints from judges and economies. As the resources are limited, the organizers seek for automation that can produce high-quality assignments under a tight schedule.

To eliminate manual and inefficient processes and improve the quality of assignments, we introduce an automated solution to generate judge assignments using weighted constraint satisfaction [9]. We identify a number of *global cost functions* [1, 5] in the problem model, which help capture all the restriction and preference requirements much more succinctly. More importantly, global cost functions provide much stronger propagation. We also benefited further by conjoining global cost functions into a single global cost function, which utilizes the flow algorithm from Régin [8]. During the assignment process, we can deliver a new assignment within a few minutes every time requirements are changed. With our automation, judge assignments can be finalized in two weeks.

The rest of the paper is arranged as follows. Section 2 further describes the scenario. We give the current practice in Section 3, which also explains our choice of solving techniques. Section 4 analyzes the problem and lists all the constraints and preferences. Section 5 gives the corresponding WCSP model, and shows how global cost functions can be conjoined in this problem. Section 6 shows that conjoining global cost functions can give optimal solutions in a few minutes. Section 7 discusses the consequence after introducing automated approaches. We conclude the paper in Section 8.

2 Problem Description

The APICTA Awards is an international awards program organized by APICTA. The competition is divided into 16 categories, ranging from School Project to Industry Application. Each category opens for candidates in 13 economies to join. Each economy nominates at most three entries in each category. Nominated candidates travel to the economy hosting the Awards, and present their

[4] `http://www.apicta2013.com/`

ICT solutions to a panel of judges specialized in the corresponding category. The judge panel picks the award winners of each category from all candidates according to their innovation and quality of work. The assessment of each category usually takes one day, but some categories require two days due to a large number of entries.

The hosting economy of APICTA Awards needs to plan for the presentation schedule for each nominated candidate and the assignments of categories for each nominated judge. The presentation schedule is constrained by the availability of presentation rooms and the number of entries in each category. In APICTA 2013, the presentation schedule had been planned by the organizers and given as parts of the input to the judge assignment process.

To ensure a fair and equitable judging process, the organizers follow a set of procedures to assign judges. Each economy has nominated at most five specialists or experts in ICT fields as judges of the APICTA Awards. The chief judge panel examines their qualifications and finalizes a list of eligible judges. In the APICTA Awards 2013, 61 judges from 13 economies are eligible. Each judge has a set of declared specialties, which corresponds to the categories in the APICTA Awards. Some judges have experience in judging the APICTA Awards before but some are new. The judge assignment process begins by assigning judges into different categories according to their specialties and experiences. Judges comment on the assignments and the organizers make modifications accordingly. This process iterates until all judges are satisfied. After the judge assignments have been confirmed, the chief judge and advisory judge panels choose a head judge for each category. The whole process is completed after all head judges are chosen. The assignments will then be announced to the public along with the presentation schedule. Therefore, the process must be completed within one month before the APICTA Events. However, the assignments usually go through a series of modifications due to negotiations and political arguments. An automated solution is desirable to ensure the judge panels can be formed on time.

The assignment of judge panels must satisfy the following criteria.

Size Restriction Each panel must be approximately the same size, consisting of 3 to 5 judges. Larger judge panels are preferred.

Maintain fairness To ensure no domination of an economy, judges in the same panel must come from different economies. The assignment must also minimize chances that judges assess entries from their own economies, which are unavoidable but give an impression of conflicts during evaluation.

Balanced workload As most judges are sponsored by APICTA, the assignment must ensure a reasonable amount of workloads to every judge. No judges should be left behind, and no judges should be in more than one panel in a day of the APICTA Events. To avoid work overload, if judges are in a category spanning two days, they should not serve in other categories throughout the Events.

Respect expertise To ensure professional judgments, no judges should be assigned to categories outside of their declared expertises.

Ensure experienced leaders Each judge panel must have at least one judge that had been a head judge before, so that their experience can be passed to inexperienced judges. More experienced judges in each panel are preferred.

Political correctness Due to political issues, judges from some economies should not be placed together in the same categories.

Allow partial assignments To increase flexibility, the organizers can force or avoid certain judges to be in a particular category.

3 Current Practice versus Constraint Programming

In the past 12 years, the judge assignments were done purely by hand following hunches. Since the criteria and objectives are not laid down explicitly, the process was far from transparent and the initial results were always complained by judges and economy leaders. The results were revised purely by hand and iterated for numerous times until (forced) consensus were reached. The processes were tedious and inefficient, especially when there were always many modifications due to negotiations and political arguments, but the resources were limited.

We propose a CP approach to develop the optimization engine in solving the judge assignment problem. A key advantage of CP is the separation of concerns in modeling and solving. Modeling involves determination of variables, domains, constraints, and objective functions. The rich constraint language allows for the model being relatively close to problem statements, making the model easy to verify and adapt. Indeed, after we delivered the first prototype, the hosting organization proposed various changes in problem statements before the assignment was finalized, and we could deliver a new solution by simply changing the model but not the implementation of solvers.

A number of attempts were tried before we arrived at the final approach and model. Instead of traditional constraint optimization, we chose the *weighted constraint satisfaction problem* (WCSP) framework after we analyzed the problem. Due to its optimization nature, the problem contains not just hard constraints but also a number of preferences. The "soft-as-hard" approach [7] in traditional constraint optimization is weak when compared with WCSP framework, which specialized in modeling preferences. As shown by Lee and Leung [5], the strong $\varnothing$-inverse consistency [5], which is a weak consistency in WCSP, is stronger in propagation than the "soft-as-hard" approach [7] in constraint optimization. We also identified global cost functions in the problem. Global cost functions in WCSP provide not just a simple language to express complex ideas, but also help increase propagation power. However, the native WCSP model failed to solve the problem within a day. We re-modeled the problem by grouping multiple global cost functions as a single one. The result could be found in a few hours, still far from being practical. We further conjoined more global cost functions as a single one. We show that the conjoined global cost function can utilize the flow algorithm from Régin [8]. Eventually, we manage to deliver a new solution within a few minutes even if the problem statements are changed. The assignment could be finalized after two weeks of blood and sweat.

4 Domain Analysis

In the following, we formally analyze the judge assignment problem and identify the corresponding constraints and objectives. The APICTA Events are hosted during $DAY = \{day_1, \ldots, day_P\}$. In APICTA 2013, $P = 2$. In day_j, a set of categories $CAT_j \subseteq CAT$, where $CAT = \{cat_1, \ldots, cat_M\}$ and $M = 16$ in APICTA 2013, are judged. Each category cat_k will have a set of entries, denoted by $entry(cat_k)$, for judges to assess. We represent a judge as $jud_i \in JUD$, where $JUD \in \{jud_1, \ldots, jud_N\}$ and $N = 61$ in APICTA 2013. Each judge jud_i is nominated from the economy $From(jud_i) \in ECO$, where $ECO = \{eco_1, \ldots, eco_Q\}$ and $Q = 13$ in APICTA 2013. Each judge $jud_i \in JUD$ had also declared the specialties defined as $SP_i \subseteq CAT$. The task at hand is to find out a time table TT, where $TT_{i,j}$ represents the set of categories assigned to the judge $jud_i \in JUD$ at $day_j \in DAY$, subject to a set of hard constraints and preferences.

4.1 Hard Constraints

The judge assignment must obey the following constraints.

Constraints on judge panel sizes

1. The size of each judge panel is at most 5 and at least 3.

Constraints on judge attendance

2. Each judge jud_i can only be in at most 1 judge panel in each day.
3. Each judge jud_i must be in at least 1 judge panel throughout the event.
4. To ensure fairness of judging, if jud_i and jud_j are in the same judge panel, they cannot be from the same economy, *i.e.* $From(jud_i) \neq From(jud_j)$.

Constraints on cross-day categories A *cross-day category* $cat_i \in CAT_{cross}$ is one spanning across two days due to a large number of entries. In APICTA 2013, 3 out of 17 are cross-day categories.

5. Judges in a cross-day category cannot be in another category, and;
6. Each judge can only be in at most 1 judge panel of a cross-day category.

Constraints on experienced judges

7. A judge jud_i is a *previous head judge*, *i.e.* $jud_i \in JUD_{head} \subseteq JUD$ iff jud_i was a head judge in APICTA before APICTA 2013. Each judge panel of a category must have at least 1 previous head judge.

Constraints on judge placement

8. Given two specific economies $eco_X \in ECO$ and $eco_Y \in ECO$, where $eco_X \neq eco_Y$. Judges from eco_X and eco_Y cannot be placed in the same panel.

Constraints on specialties

9. A judge jud_i is in category cat_k iff cat_k is a specialty of jud_i, *i.e.* $cat_k \in SP_i$.

Constraints on pre-setting We define, for each judge $jud_i \in JUD$, $Assign_i \subseteq SP_i$ to be the categories that jud_i must be in, and $Avoid_i \subseteq SP_i$ to be the categories that jud_i must not be in. In APICTA 2013, $|Assign_i| \leq 1$ and $|Avoid_i| \leq 1$ for every judge $jud_i \in JUD$.

10. A judge jud_i is in category $cat_k \in Assign_i$ iff $Assign_i \neq \varnothing$, and;
11. A judge jud_i is not in category $cat_k \in Avoid_i$ iff $Avoid_i \neq \varnothing$.

Constraint on judge workload

12. The *workload* of a judge jud_i in a category cat_k is the number of entries in cat_k. All judges evaluate at least 7 entries in total.

4.2 Preferences

The objective is to minimize the weighted sum of the following preference functions. The weights determine the importance of each preference, *i.e.* the most important one will have the highest weight. We adjusted the weights through experiments.

Preference on conflicts Define a *conflict* as a function $conf : JUD \times CAT \mapsto \mathbb{N}$, which returns the number of entries in cat_k from $From(jud_i)$, *i.e.* the same economy as jud_i.

13. Minimize the total number of conflicts, *i.e.*

$$\min \sum_{jud_i \in JUD} \sum_{day_j \in DAY} \sum_{cat_k \in TT_{i,j}} conf(jud_i, cat_k)$$

Preference on maximizing judge panel sizes The assignment prefers larger judge panels,and penalize the judge panels with size less than 5.

14. Minimize the total penalties due to small panel sizes, *i.e.*

$$\min \sum_{day_j \in DAY} \sum_{cat_k \in CAT_j} (5 - |\{jud_i \mid cat_k \in TT_{i,j}\}|)$$

Preference on maximizing experienced share A judge $jud_i \in JUD_{exp}$ is *experienced* iff jud_i has judged in APICTA before. Note that $JUD_{head} \subseteq JUD_{exp}$. More experienced judges in each panel are preferred, and penalize the judge panels with number of experienced judges less than 5.

15. Minimize the total penalties due to less experienced judges in panels, *i.e.*

$$\min \sum_{day_j \in DAY} \sum_{cat_k \in CAT_j} (5 - |\{jud_i \mid cat_k \in TT_{i,j} \wedge jud_i \in JUD_{exp}|)$$

5 Problem Modeling

We first give a background on weighted constraint satisfaction problems (WCSP) and global cost functions. While there are many ways of formulating the judge assignment problem into WCSP, we give the one that allows natural expression of cost functions and utilizes global cost functions. Based on the model, we further propose how the global cost functions can be combined to give stronger propagation power.

5.1 Weighted Constraint Satisfaction and Global Cost Functions

A *WCSP* [9] is a tuple $(\mathcal{X}, \mathcal{D}, \mathcal{C}, \top)$. $\mathcal{X}$ is a set of variables $\{x_1, x_2, \ldots, x_n\}$. Each variable has its finite domain $D(x_i) \in \mathcal{D}$ containing possible values for x_i. A tuple $\ell \in \mathcal{L}(S) = D(x_{s_1}) \times \ldots \times D(x_{s_n})$ is used to represent an assignment on $S = \{x_{s_1}, \ldots, x_{s_n}\} \subseteq \mathcal{X}$. The notation $\ell[x_i]$ denotes the value assigned to x_i in ℓ, and $\ell[S']$ denotes the tuple formed from projecting ℓ onto $S' \subseteq S$. $\mathcal{C}$ is a set of cost functions. Each cost function $W_S \in \mathcal{C}$ has its scope $S \subseteq \mathcal{X}$, and maps $\ell \in \mathcal{L}(S)$ to a cost in the valuation structure $V(\top) = ([0 \ldots \top], \oplus, \leq)$. $V(\top)$ contains a set of integers $[0 \ldots \top]$ with standard integer ordering $\leq$. $\top$ is a finite or infinite integer corresponding to forbidden assignments. Addition $\oplus$ is defined by $a \oplus b = \min(\top, a + b)$. Subtraction $\ominus$ is defined only for $a \geq b$, $a \ominus b = a - b$ if $a \neq \top$ and $\top \ominus a = \top$ for any a. The *cost* of a tuple $\ell \in \mathcal{L}(\mathcal{X})$ in a WCSP is defined as $cost(\ell) = \bigoplus_{W_S \in \mathcal{C}} W_S(\ell[S])$. A tuple ℓ is an optimal valid *solution* of a WCSP if $cost(\ell)$ is minimum among all tuples in $\mathcal{L}(\mathcal{X})$ and $cost(l) < \top$.

A *global cost function* [1,5] is a cost function with special semantics, based on which efficient algorithms can be designed for consistency enforcements. In particular, we denote a global cost function as $\textsc{Soft\_GC}^{\mu}_{m}(S)$ if it is derived from the corresponding hard global constraint GC with variable scope S, a violation measure μ, and a weight constant m. The cost function $\textsc{Soft\_GC}^{\mu}_{m}(S)$ returns $m \cdot \mu(\ell)$ to indicate how much a tuple $\ell \in \mathcal{L}(S)$ has violated GC, or 0 if the tuple satisfies GC. Two examples of violation measures for global constraints involves counting are μ_{var} and μ_{val}: $\textsc{Soft\_GC}^{var}_{1}(S)$ returns the minimum number of assignments modified to satisfy GC [7]; while $\textsc{Soft\_GC}^{val}_{1}(S)$ returns the number of values exceeding the boundaries allowed by GC [11]. We assume $m = 1$ if m is not specified. If $m = \top$, a global cost function represents a **hard** constraint.

5.2 Problem Formulation

Define $P = (\mathcal{X}, \mathcal{D}, \mathcal{C}, \top)$ to be the WCSP model for the judge assignment problems. The variable $x_{i,j} \in \mathcal{X}$ gives the category that jud_i assesses on day_j. The domain $D(x_{i,j})$ of each variable $x_{i,j}$ is the set of categories $CAT_j \subseteq CAT$ judged on day_j, with a dummy category cat_0 to indicate no judging on a day. Each constraint and preference are enforced as follows.

Constraints on judge panel sizes Constraint 1 can be enforced by the global cost function $\textsc{soft\_among}^{var}(S, lb, ub, V)$, which returns $max(0, lb - t(\ell), t(\ell) - ub)$ for each tuple $\ell \in \mathcal{L}(S)$, where $t(\ell) = |\{x_i \in S \mid \ell[x_i] \in V\}|$ [10]. For each $day_j \in DAY$, we place one $\textsc{Soft\_Among}^{var}_{\top}(\mathcal{X}_j, 3, 5, \{cat_i\})$ for each $cat_i \in CAT_j$, where $\mathcal{X}_j = \{x_{i,j} \mid jud_i \in JUD\}$.

Constraints on judge attendance Constraint 2 is always satisfied for all valid assignments. Constraint 3 can be enforced by placing one $\textsc{Soft\_among}^{var}_{\top}(\{x_{i,j} \mid day_j \in DAY\}, 1, |DAY|, CAT)$ for each jud_i. As $|DAY| = 2$, binary table cost functions were used instead. Constraint 4 can be enforced by the global cost functions $\textsc{Soft\_GCC}^{val}(S, LB, UB)$ [11]. Given $\Sigma = \bigcup_{x_i \in S} D(x_i)$. Define the

number of occurrences of a value $v \in \Sigma$ in ℓ by $\#(\ell, v)$ and two functions $s(\ell, v)$ and $e(\ell, v)$ as follows.

$$s(\ell,v) = \begin{cases} LB(v) \ominus \#(\ell,v), & \text{if } \#(\ell,v) \leq LB(v) \\ 0, & \text{otherwise} \end{cases} \quad e(\ell,v) = \begin{cases} \#(\ell,v) \ominus UB(v), & \text{if } \#(\ell,v) \geq UB(v) \\ 0, & \text{otherwise} \end{cases}$$

The global cost function $\textsc{Soft\_GCC}^{val}$ is defined for each tuple $\ell \in \mathcal{L}(S)$ as follows [11].

$$\textsc{Soft\_GCC}^{val}(S, LB, UB)(\ell) = \bigoplus_{v \in \Sigma} s(\ell, v) \oplus \bigoplus_{v \in \Sigma} e(\ell, v)$$

Define $\{\mathcal{X}_{j,eco_t}\}$ to be the partition of $\mathcal{X}$ according to the day of the events and the economies the judges represent, *i.e.* $x_{i,j} \in \mathcal{X}_{j,eco_t}$ iff $From(jud_i) = eco_k$. For each day_j and economy eco_t, we place one $\textsc{Soft\_GCC}^{val}_{\top}(\mathcal{X}_{j,eco_t}, LB, UB)$, which $LB(cat_k) = 0$ for all $cat_k \in CAT_j \cup \{cat_0\}$ and UB are defined as follows.

$$UB(cat_k) = \begin{cases} 1, & k \neq 0, \\ |\mathcal{X}|, & \text{otherwise.} \end{cases} \tag{1}$$

Constraints on cross-day categories Constraints 5 and 6 can be enforced by the binary cost function *JoinOnlyCrossCat* placed on each pair of variables $\{(x_{i,h}, x_{i,k}) \mid h \neq k \wedge day_h, day_k \in DAY\}$. The cost function is defined for each pair of values (v_h, v_k), where $v_h \in D(x_{i,h})$ and $v_k \in D(x_{i,k})$ as follows.

$$JoinOnlyCrossCat_{\{x_{i,h}, x_{i,k}\}}(v_h, v_k) = \begin{cases} 0, & \text{if } v_h \notin CAT_{cross} \text{ and } v_k \notin CAT_{cross} \\ 0, & \text{if } v_h \in CAT_{cross} \text{ and } v_k = v_h \\ \top, & \text{otherwise} \end{cases}$$

Constraints on experienced judges Constraint 7 enforced by placing one $\textsc{Soft\_GCC}^{val}_{\top}(\{x_{i,j} \mid jud_i \in JUD_{head}\}, LB, UB)$ for each $day_j \in DAY$, where LB and UB are defined as follows.

$$LB(cat_j) = \begin{cases} 1, & j \neq 0, \\ 0, & \text{otherwise.} \end{cases} \quad UB(cat_j) = \begin{cases} 1, & j \neq 0, \\ |\mathcal{X}|, & \text{otherwise.} \end{cases}$$

Constraints on judge placement Constraint 8 can be fused in constraint 4 by considering eco_X and eco_Y as a single economy.

Constraints on specialties Constraint 9 can be enforced by placing the unary cost function *Specialty* on each variable $x_{i,j} \in \mathcal{X}$. The function *Specialty* is defined for each value $v \in D(x_{i,j})$ as follows.

$$Specialty_{\{x_{i,j}\}}(v) = \begin{cases} 0, & \text{if } v = cat_0 \text{ or } v \in SP_i \\ \top, & \text{otherwise} \end{cases}$$

Constraints on pre-setting Constraints 10 and 11 can be enforced by the unary cost functions *Set* and *Unset* placed on each variable $x_{i,j} \in \mathcal{X}$, defined for each value $v \in D(x_{i,j})$ as follows.

$$Set_{\{x_{i,j}\}}(v) = \begin{cases} 0, & \text{if } v \in Assign_i \\ & \text{or } Assign_i = \varnothing; \\ \top, & \text{otherwise.} \end{cases} \quad Unset_{\{x_{i,j}\}}(v) = \begin{cases} \top, & \text{if } v \in Avoid_i \\ & \text{and } Avoid_i = \varnothing; \\ 0, & \text{otherwise.} \end{cases}$$

Constraint on judge workload Define $entry(cat_0) = 0$. Since $|DAY| = 2$, constraint 12 can be simply enforced by the binary cost function $WLLimit$ placed on each pair of variables $\{(x_{i,h}, x_{i,k}) \mid h > k \wedge day_h, day_k \in DAY\}$. The cost function is defined for each pair of values (v_h, v_k), where $v_h \in D(x_{i,h})$ and $v_k \in D(x_{i,k})$, as follows.

$$WLLimit_{\{x_{i,h},x_{i,k}\}}(v_h, v_k) = \begin{cases} \top, & \text{if } entry(v_h) + entry(v_k) < 7 \text{ if } v_h \neq v_k, \text{ or} \\ & entry(v_h) < 7 \text{ if } v_h = v_k; \\ 0, & \text{otherwise} \end{cases}$$

Preference on conflicts We define the weight of preference 13 as $\epsilon_{conflict}$. Preference 13 can be enforced by the unary cost function $Conflict$ placed on each variable $x_{i,j} \in \mathcal{X}$, which $Conflict_{\{x_{i,j}\}}(v) = \epsilon_{conflict} \cdot conf(jud_i, v)$ for every $v \in D(x_{i,j})$.

Preference on maximizing judge panel size We observe that preference 14 is a special case of Soft_Among^{var}. We define the weight of preference 14 as $\epsilon_{maxsize}$. One Soft_Among$^{var}_{\epsilon_{maxsize}}(\mathcal{X}_j, 5, 5, \{cat_k\})$ is posted for each day_j and each category cat_k.

Preference on maximizing experienced share Define the weight of preference 15 as ϵ_{exp}. One Soft_Among$^{var}_{\epsilon_{exp}}(\{x_{i,j} \mid jud_i \in JUD_{exp}\}, 5, 5, \{cat_k\})$ is posted for each day_j and each category cat_k.

5.3 Conjoining Cost Functions

As the original WCSP model cannot be solved within a day, we consider conjoining global cost functions to further reduce the search space. Lee *et al.* [6] give a general technique on conjoining global cost functions using linear programs, and show that enforcing consistencies on a conjoined cost function is stronger than enforcing the same consistencies on separate cost functions. In this section, we give a special case on conjoining global cost functions using *flow networks*.

A *flow network* $G = (V, E, w, c, d)$ is a connected directed graph (V, E), in which each edge $e \in E$ has a weight w_e, a capacity c_e, and a demand $d_e \leq c_e$. An (s,t)-*flow* f from a source $s \in V$ to a sink $t \in V$ of a value $value(f)$ in G is defined as a mapping from E to real numbers such that:

- $\sum_{(s,u)\in E} f_{(s,u)} = \sum_{(u,t)\in E} f_{(u,t)} = value(f)$;
- $\sum_{(u,v)\in E} f_{(u,v)} = \sum_{(v,u)\in E} f_{(v,u)} \; \forall \; v \in V \setminus \{s,t\}$;
- $d_e \leq f_e \leq c_e \; \forall \; e \in E$.

For simplicity, we call an (s,t)-flow as a flow if s and t have been specified. The *cost* of a flow f is defined as $cost(f) = \sum_{e\in E} w_e f_e$. A *minimum cost flow* problem of a value α is to find the flow f of $value(f) = \alpha$ such that its cost is minimum. If α is not given, it is assumed to be the maximum value among all flows.

A global cost function W_S is *flow-based* if W_S can be represented as a flow network $G = (V, E, w, c, d)$ such that $min\{W_S(\ell) \mid \ell \in \mathcal{L}(S)\} = min\{cost(f) \mid f \text{ is the maximum } (s,t)\text{-flow of } G\}$, where $s \in V$ is the fixed source and $t \in V$ is the fixed destination. One example of flow-based global cost function is $\textsc{Soft\_GCC}^{val}$ [11].

The global cost functions $\{\textsc{Soft\_Among}^{var}(\mathcal{X}_j, lb_i, ub_i, \{cat_k\}) \mid cat_k \in CAT\}$ in constraint 1, preferences 14 and 15 can be conjoined respectively as a single $\textsc{Soft\_GCC}^{val}$ for each $day_j \in DAY$. We show as follows.

Proposition 1. *Given a set* $\{\textsc{Soft\_Among}^{var}(S, lb_i, ub_i, \Omega_i) \mid i = 1 \dots h\}$. *If* $\Omega_i \cup \Omega_j = \varnothing$ *for* $i \neq j$ *and* $|\Omega_i| = 1$ *for every* i, *the following holds for every tuple* $\ell \in \mathcal{L}(S)$:

$$\textsc{Soft\_GCC}^{val}(S, LB, UB)(\ell) = \bigoplus_{i=0}^{m} \textsc{Soft\_Among}^{var}(S, lb_i, ub_i, \Omega_i)(\ell)$$

The lower bound LB *is defined as* $LB(v) = lb_i$ *iff* $v \in \Omega_i$, *and the upper bound* UB *is defined as* $UB(v) = ub_i$ *iff* $v \in \Omega_i$.

Proof. Define $v_i \in \Omega_i$. By definitions, for every tuple $\ell \in \mathcal{L}(S)$, $max(0, lb_i - t(\ell), t(\ell) - ub_i) = s(\ell, v_i) \oplus e(\ell, v_i)$. Results follow. □

By conjoining $\textsc{Soft\_Among}^{var}$ into $\textsc{Soft\_GCC}^{val}$, the problem instance can be solved within a few hours, but is still far from being practical. We further conjoin $\textsc{Soft\_Among}^{var}$ in constraint 1 and preference 14 and $\textsc{Soft\_GCC}^{val}$ in constraints 4 and 8 into a single global cost function $\textsc{Soft\_GCC\_Among}^{val+bvar}$. We found that the $\textsc{Soft\_GCC\_Among}^{val+bvar}$ is *flow-based*, allowing consistencies in WCSP to be enforced by flow networks.

The cost functions $\{\textsc{Soft\_Among}^{var}_{\top}(\mathcal{X}_j, 3, 5, \{cat_k\}) \mid cat_k \in CAT\}$ in constraint 1 and $\{\textsc{Soft\_Among}^{var}(\mathcal{X}_j, 5, 5, \{cat_k\}) \mid cat_k \in CAT\}$ in preference 14 can be conjoined into a single $\textsc{Soft\_Among}^{bvar}(\mathcal{X}_j, 3, 5, CAT)$ for each $day_i \in DAY$. The new violation measure μ_{bvar} forbids the number of specified values exceeding the boundaries given by cost functions, and favor the tuple containing more specified values. The cost function $\textsc{Soft\_Among}^{bvar}(S, lb, ub, \Omega)$ is defined for each $\ell \in \mathcal{L}(S)$ as follows.

$$\textsc{Soft\_Among}^{bvar}(S, lb, ub, \Omega) = \begin{cases} ub \ominus t(\ell), & \text{if } t(\ell) \geq lb \text{ and } t(\ell) \leq ub \\ \top, & \text{otherwise} \end{cases}$$

We further conjoin $\textsc{Soft\_Among}^{bvar}$ and $\textsc{Soft\_GCC}^{val}$ into $\textsc{Soft\_GCC\_Among}^{val+bvar}$. The violation measure $\mu_{val+bvar}$ is a conjoined violation measure from μ_{bvar} and μ_{val} used by $\textsc{Soft\_Among}^{bvar}$ and $\textsc{Soft\_GCC}^{val}$ respectively. Given a set of global cost functions $\mathcal{C}_{GCC} = \{\textsc{Soft\_GCC}^{val}(S_i, LB_i, UB_i) \mid i = 1, \dots, m\}$ and $\mathcal{C}_{Among} = \{\textsc{Soft\_Among}^{bvar}(K_j, lb_j, ub_j, \Omega_j) \mid j = 1, \dots, h\}$, the cost function $\textsc{Soft\_GCC\_Among}^{val+bvar}(\{(S_i, UB_i, LB_i)\}\{(K_j, lb_j, ub_j, \Omega_j)\})$ is

defined as a global cost function formed by conjoining $\mathcal{C}_{GCC}$ and $\mathcal{C}_{Among}$, *i.e.* for every tuple $\ell \in \mathcal{L}(\bigcup_{i=1}^{m} S_i \cup \bigcup_{j=1}^{h} K_j)$:

$$\textsc{Soft\_GCC\_Among}^{val+bvar}(\ell) = \bigoplus_{i=1}^{m} \textsc{Soft\_GCC}^{val}(S_i, LB_i, UB_i)(\ell[S_i]) \oplus \bigoplus_{j=1}^{h} \textsc{Soft\_Among}^{bvar}(K_j, lb_j, ub_j, \Omega_j)(\ell[K_j])$$

We show $\textsc{Soft\_GCC\_Among}^{val+bvar}$ is flow-based as follows.

Theorem 1. *The cost function* $\textsc{Soft\_GCC\_Among}^{val+bvar}(\{(S_i, LB_i, UB_i)\}, \{(K, lb_j, ub_j, \Omega_j)\})$ *is flow-based if the following condition holds.*

- $S_i \cap S_j = \varnothing$ *if* $i \neq j$;
- $LB(v) = 0$ *for* $v \in \Sigma_i$, *where* $\Sigma_i = \bigcup_{x_j \in S_i} D(x_j)$, *for each* $i = 1, \ldots, m$;
- $\Omega_i \cap \Omega_j = \varnothing$ *if* $i \neq j$, *and;*
- $K = \bigcup_{i=1}^{m} S_i$ *for each* $j = 1, \ldots, h$.

Proof. Without loss of generality, we assume $\Sigma = \bigcup_{j=1}^{h} \Omega_j$. If there exists a value $u \in \bigcup_{j=1}^{h} \Omega_j$ that does not exist in Σ, we can add a dummy $\textsc{Soft\_Among}^{var}(K, 0, |K|, \{u\})$ into the set of cost functions.

The flow network can be constructed using the method suggested by Régin [8]. We construct a single flow network $G = (V, E, w, c, d)$ representing a set of $\textsc{Soft\_GCC}^{val}$ and $\textsc{Soft\_Among}^{var}$ as follows.

- $V = K \cup \{v_{iv} \mid v \in \Sigma_i\} \cup \{\mu_j \mid j = 1, \ldots, h\} \cup \{s, t\}$;
- $E = A_s \cup A_K \cup A_v \cup A_t \cup A_{gcc-ex} \cup A_{among-short} \cup A_{among-vio} \cup A_{among-ex}$, where:
  - $A_s = \{(s, x_j) \mid x_j \in \mathcal{X}\}$;
  - $A_K = \{(x_j, v_{iu}) \mid x_j \in \mathcal{X}_i \wedge u \in D(x_j)\}$;
  - $A_v = \{(v_{iu}, \mu_j) \mid u \in V_j \wedge j = 1, \ldots, h\}$;
  - $A_t = \{(\mu_j, t) \mid j = 1, \ldots, h\}$;
  - $A_{gcc-ex} = \{(v_{iu}, \mu_j) \mid u \in \Sigma_i \wedge u \in \Omega_j\}$;
  - $A_{among-vio} = \{(s, \mu_j) \mid j = 1, \ldots, h\}$;
  - $A_{among-short} = \{(s, \mu_j) \mid j = 1, \ldots, h\}$, and;
  - $A_{among-ex} = \{(\mu_j, t) \mid j = 1, \ldots, h\}$.
- $c_e = \begin{cases} UB_i(u), & \text{if } e = (v_{iu}, \mu_j) \in A_v \\ ub_j, & \text{if } e = (\mu_j, t) \in A_t \\ ub_j \ominus lb_j, & \text{if } e = (s, \mu_j) \in A_{among-short} \\ |K|, & \text{if } e \in A_{gcc-ex} \cup A_{among-ex} \\ 1, & \text{otherwise} \end{cases}$
- $d_e = \begin{cases} ub_j, & \text{if } e = (\mu_j, t) \in A_t \\ 0, & \text{otherwise} \end{cases}$
- $w_e = \begin{cases} \top, & \text{if } e \in A_{among-vio} \cup A_{among-ex} \\ 1, & \text{if } e \in A_{gcc-ex} \cup A_{among-short} \\ 0, & \text{otherwise} \end{cases}$

In the flow network, there may exist multiple edges between two nodes. The edges A_v enforce Soft_GCC^{val} while A_{gcc-ex} give the corresponding violation cost. The edges A_t enforce Soft_Among^{bvar} while $A_{among-short}$ give the corresponding violation cost. The edges $A_{among-ex}$ and $A_{among-vio}$ ensure all tuples have corresponding maximum flows.

Using the similar reasoning as in Proposition 7 given by Régin [8], a maximum (s,t)-flow in G corresponds to an assignment to K. With the similar reasoning by van Hoeve [11], the minimum cost of maximum flows corresponds to the minimum of Soft_GCC_Among$^{val+bvar}$. Results follow. □

An example of the flow network G is shown in Figure 1, based on a set of variables $S = \{x_i \mid i = 1, \ldots 5\}$ and $D(x_i) = \{cat_1, cat_0\} \subseteq CAT$. The Soft_GCC_Among$^{val+bvar}$ consists of the following cost functions.

- Soft_Among$^{bvar}(\{x_1, x_2, x_3, x_4, x_5\}, 2, 5, \{cat_1\})$
- Soft_Among$^{bvar}(\{x_1, x_2, x_3, x_4, x_5\}, 0, 5, \{cat_0\})$
- Soft_GCC$^{val}(\{x_1, x_2\}, LB, UB)$, and;
- Soft_GCC$^{val}(\{x_3, x_4, x_5\}, LB, UB)$, where LB and UB are defined as in constraint 4.

The pair of numbers on the edges represent the demands and capacities of the edges. If no numbers are on the edge, the edge has zero demand and unit capacity. If the edge is dotted, the edge has unit weight. Otherwise, the edge has zero weight. For simplicity, we omit the edges with weight equal to $\top$. The thick lines show the flow corresponding to the tuple $\ell = (cat_1, cat_1, cat_0, cat_0, cat_0)$ having a cost of 4: 1 from Soft_GCC^{val} and 3 from Soft_Among^{bvar}.

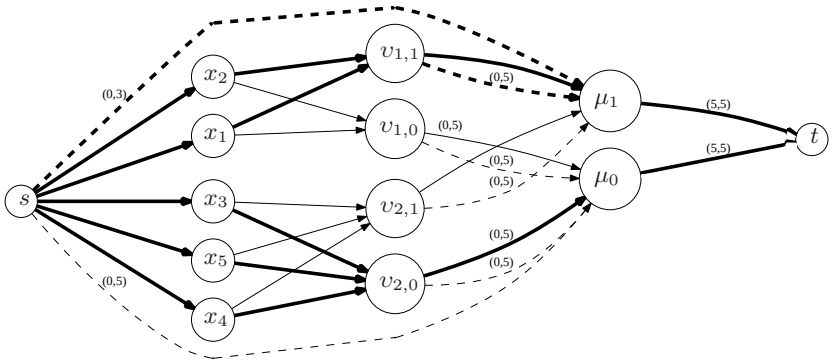

Fig. 1. An example of the flow network

We further show that the conjoined cost function is *flow-based projection-safe* [5]. If the cost function is flow-based projection-safe, stronger consistencies like GAC* [2,5], FDGAC* [5], and weak EDGAC* [5] can be enforced in polynomial time throughout the search. Stronger consistencies help remove more search

places, and reduce the runtime if the reduction in search space can compensate the time on enforcing consistencies.

A global cost function W_S is *flow-based projection-safe* [5] iff W_S is flow-based, and for all W'_S derived from W_S by a series of projections and extensions, W'_S is flow-based. Lee and Leung [5] give sufficient conditions on flow-based projection-safety.

1. W_S is flow-based, with the corresponding network $G = (V, E, w, c, d)$ with a fixed source $s \in V$ and a fixed destination $t \in V$;
2. there exists a subjective function mapping each maximum flow f in G to each tuple $\ell_f \in \mathcal{L}(S)$, and;
3. there exists an injection mapping from an assignment (x_i, v) that set the variable x_i to $v \in D(x_i)$ to a subset of edges $\bar{E} \subseteq E$ such that for all maximum flow f and the corresponding tuple ℓ_f, $\sum_{e \in \bar{E}} f_e = 1$ whenever $\ell_f[x_i] = v$, and $\sum_{e \in \bar{E}} f_e = 0$ whenever $\ell_f[x_i] \neq v$

We show that Soft_GCC_Among$^{val+bvar}$ is flow-based projection-safe as follows.

Theorem 2. *If the cost function* Soft_GCC_Among$^{val+bvar}(\{(S_i, UB_i, LB_i)\}, \{(K_j, lb_j, ub_j, \Omega_j)\})$ *satisfies the conditions stated in Theorem 1, it is flow-based projection-safe.*

Proof. Theorem 1 already shows the Soft_GCC_Among$^{val+bvar}$ satisfies conditions 1 and 2. In addition, the edge (x_j, v_{iu}) corresponds to assigning u to x_j. Results follow. □

The cost functions in constraints 1, 4, 8, and preference 14 satisfies the conditions given in Theorem 1 if they are grouped by $day_j \in DAY$. The conjoined cost function shown below, defined for $day_j \in DAY$, is flow-based projection-safe.

$$\text{Soft\_GCC\_Among}^{val+bvar}\ (\ \{(X_{j,eco_t}, LB, UB) \mid eco_t \in ECO\},\ \{(X_j, 3, 5, \{cat_i\}) \mid cat_i \in CAT\})$$

By applying the above conjoined cost functions, the assignments can be found within a few minutes, even if the requirements are changed frequently. In the following, we show the robustness of our solution by experiments.

6 Experiments

As the data from previous years are not available, the experiments are purely based on the data from APICTA 2013 and conducted on Toulbar2 [5], an open-source WCSP solver. In the experiments, variables are assigned in lexicographic order. Value assignments start with the values having the minimum unary costs. Weak EDGAC* [5] is enforced during search with no initial upper bound. Each

[5] `https://mulcyber.toulouse.inra.fr/projects/toulbar2/`

test is conducted on a Linux Cluster (4 × 2GHz CPU) machine with 3GB memory. In all experiments, we set $\epsilon_{conflict} = 1$, $\epsilon_{maxsize} = 200$, and $\epsilon_{exp} = 50$.

We first compare the runtime in solving the APICTA 2013 instance using different models. The solver cannot give the optimal for the native model as stated in Section 5.2 after 3 days of execution. The model applying Proposition 1 can be solved after 5.3 hours. With SOFT_GCC_AMONG$^{val+bvar}$, the solver gives the optimal solution in 107.6 seconds.

We further show the robustness of our solution by simulating the judge assignment process using the conjoined model. We mark the instance from APICTA 2013 as instance 0. Each instance i, where $i > 0$, is modified from instance 0. We keep around 10% of judge assignments in the solution of instance 0 as the preset assignments of instance i. We randomly modify the requirements of the remaining 90% of judges on top of instance i as follows.

- Remove a judge and add a new judge from a different economy;
- Modify the specialties of a judge;
- Avoid a judge to be in a specific category, and;
- Withdraw some entries so that the conflict is changed.

We generate 400 instances, each of which is allowed to run for 15 minutes.[6]

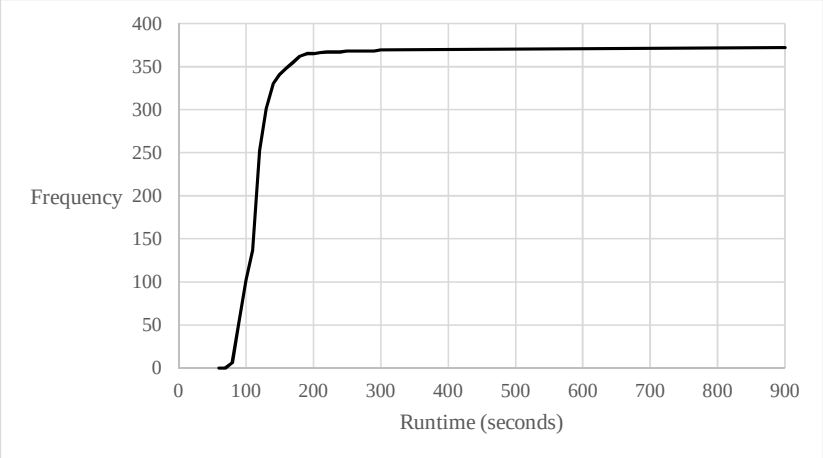

Fig. 2. Frequency distribution according to the runtime

We present the results as the frequency distribution as shown in Figure 2. The x-axis is the runtime and the y-axis gives the number of instances able to be solved within the runtime. We observe that 93% of the instances can be solved within 200 seconds, while only 29 out of 400 instances cannot be solved within 15 minutes.

[6] The solver and instances can be found online: `http://www.cse.cuhk.edu.hk/~klleung/cp14/JudgeAssign.tar.gz`

7 Discussion

After we gave an initial solution to the organizers, we were asked to do modifications, as simulated in the previous section, before finalization. The modifications were unpredictable, but we could cope with the modifications and gave an updated optimal solution within a few minutes.

The organizers also received far less requests on modifications from the judges on the assignments. Throughout the process, only one judge complained the assignments, and it was resolved by correcting the specialties.

Besides, our solution speeded up the process. The judge assignment needs to be completed one month before the APICTA event. In previous years, the assignment was completed only a few days before the deadline. With automation, the process was completed in two weeks, including endorsement from advisory judges. Compared with previous years, the time required was greatly reduced.

8 Conclusion

Judge assignment problems have been studied in the field of sport tournaments. Lamghari and Ferland [4] formulates the problems into linear programs and solves by tabu search. Fernando *et al.* [3] also use integer linear programming to compute referee assignments in football matches.

Our contributions are three-fold. First, we implement an automated solution for APICTA 2013 to generate the most preferred assignments of each judge to each category. Our solution helps lay down all restrictions and preference explicitly, and generate a new assignment within a few minutes every time the requirements and preferences are changed, shortening the process to two weeks. Second, we give a real-life example on global cost functions [1, 5] in WCSP. By using Soft_GCC^{val} [11] and Soft_Among^{var} [10], WCSP can model restrictions that involves several variables. Third, we further give special cases of conjoining a group of Soft_GCC^{val} and Soft_Among^{var} via flow networks. The original model cannot be solved in a day. We refine the model by grouping a set of Soft_Among^{var} into Soft_GCC^{val}, but still not practical. We further conjoin Soft_Among^{var} and Soft_GCC^{val} in the model as a single Soft_GCC_Among$^{val+bvar}$, which is flow-based projection-safe [5]. We show by experiments that conjoining global cost functions can solve an instance within a few minutes after requirements are modified.

We plan to refine our solution for judge assignments in APICTA 2014[7] to produce solutions in a shorter time. Eventually, we hope our technique can be applied to other similar scenarios such as banquet seating planning.

Acknowledgment

We are grateful to the anonymous referees for their constructive comments. The work described in this paper was substantially supported by grant CUHK413710

[7] `https://www.facebook.com/APICTA2014`

from the Research Grants Council of Hong Kong SAR and grant F-HK019/12T from the Consulate General of France of Hong Kong and the Research Grants Council of Hong Kong SAR.

References

1. M. Cooper, S. de Givry, M. Sanchez, T. Schiex, M. Zytnicki, and T. Werner. Soft Arc Consistency Revisited. *Artificial Intelligence*, 174:449–478, 2010.
2. M. Cooper and T. Schiex. Arc Consistency for Soft Constraints. *Artifical Intelligence*, 154:199–227, 2004.
3. A. Fernando, G. Durn, and M. Guajardo. Referee Assignment in the Chilean Football League Using Integer Programming and Patterns. *International Transactions in Operational Research*, 21(3):415 – 438, 2014.
4. A. Lamghari and J. A. Ferland. Assigning Judges to Competitions of Several Rounds Using Tabu Search. *European Journal of Operational Research*, 210(3):694 – 705, 2011.
5. J. H. M. Lee and K. L. Leung. Consistency Techniques for Global Cost Functions in Weighted Constraint Satisfaction. *Journal of Artificial Intelligence Research*, 43:257–292, 2012.
6. J. H. M. Lee, K. L. Leung, and Y. M. Shum. Consistency Techniques for Polytime Linear Global Cost Functions in Weighted Constraint Satisfaction. *CONSTRAINTS*, 19(3):270–308, 2014.
7. T. Petit, J.-C. Régin, and C. Bessière. Specific Filtering Algorithm for Over-Constrained Problems. In *Proceedings of CP'01*, pages 451–463, 2001.
8. J.-C. Régin. Combination of among and cardinality constraints. In *Proceedings of CPAIOR'05*, pages 288–303, 2005.
9. T. Schiex, H. Fargier, and G. Verfaillie. Valued Constraint Satisfaction Problems: Hard and Easy Problems. In *Proceedings of IJCAI'95*, pages 631–637, 1995.
10. C. Solnon, V. Cung, A. Nguyen, and C. Artigues. The Car Sequencing Problem: Overview of State-of-the-Art Methods and Industrial Case-Study of the ROADDEF'2005 Challege Problem. *European Journal of Operational Research*, 191(3):912–927, 2008.
11. W.-J. van Hoeve, G. Pesant, and L.-M. Rousseau. On Global Warming: Flow-based Soft Global Constraints. *J. Heuristics*, 12(4-5):347–373, 2006.