Abstract

The goal of this project is to implement constrain satisfaction model for general case of Social Golfer Problem. The model is implemented in SICStus Prolog using its CSP over Finite Domains library.

1 Problem Definition

From Wolfram Math World original definition of problem is as follows:

"Twenty golfers wish to play in foursomes for 5 days. Is it possible for each golfer to play no more than once with any other golfer? The answer is yes, and the following table gives a solution.

Mon	ABCD	EFGH	IJKL	MNOP	QRST
Tue	AEIM	BJOQ	CHNT	DGLS	FKPR
Wed	AGKO	BIPT	CFMS	DHJR	ELNQ
Thu	AHLP	BKNS	CEOR	DFIQ	GJMT
Fri	AFJN	BLMR	CGPQ	DEKT	HIOS

Event organizers for bowling, golf, bridge, or tennis frequently tackle problems of this sort, unaware of the problem complexity. In general, it is an unsolved problem. A table of known results is maintained by Harvey."

1.1 General case problem definition

Lets first define problem parameters.

Definition 1.1. (i) $N \in \mathbb{N}$ ... number of players

(ii) $K \in \mathbb{N}$ ... size of groups

(iii) $D \in \mathbb{N}$ ... number of days

(iv) $G \in \mathbb{N}$ ... number of groups

Now when we have our problem parametrized we can describe model assuming those parameters. But lets first define following simple notation.

Definition 1.2. For given $S \in \mathbb{N}$ define:

$$[S] \equiv \{0, 1, 2, ..., S-1\}.$$

Definition 1.3. Lets define groups as sets $\forall i \in [G], j \in [D] : S_{i,j}$ such as:

(i) Players are identified by numbers in given groups.

$$\forall p \in S_{i,j} : p \in [N]$$

(ii) Every group has exactly K players in it.

$$|S_{i,j}| = K$$

(iii) Every player is at most once in group. $S_{i,j}$ are sets thus this holds trivaly.

(iv) Every player is playing exactly once in every day.

$$\forall i \in [D] : \bigsqcup_{j \in [G]} S_{i,j} = [N]$$

(v) Every player plays no more then once with any other player. Which is same as every pair of players is in at most 1 group.

$$\forall x, y \in [N] : |\{S | \exists i \in [D] \wedge \exists j \in [G] : x \in S_{i,j} \wedge y \in S_{i,j}\}| < 2$$

Remark. We can see that conditions ii and iv implies that:

$$N = G * K.$$

In following section our task is going to be to choose proper representation for this model in terms of CSP.

2 CSP model representation

Simpliest representation which comes right into our mind is using directly definition 1.1.

2.1 Straight forward model

We just create for each slot in each group in each day variable and will try to assign right players.

Definition 2.1. Lets define CSP model $M_a = \langle V_a, D_a, C_a \rangle$ such as:

(i)

$$V_a = \{V_k | \forall i \in [D], \forall j \in [G], \forall p \in [K] \exists k \in \mathbb{N} : V_k \text{ is CSP variable } \wedge k = i*G*K+j*K+p\}$$

(ii)

$$D_a = \{D_i | \forall V_i \in V : D_i = [N]\}$$

(iii)

$$C_a = \{C_1, C_2, C_3, C_4\}$$

(a) To ensure conditions ii and iii we need constrain C_1 requiring that player can be assigned to at most one variable in certain group.

(b) To ensure condition iv we need constrain C_2 requiring that each player playes once per day.

(c) To ensure condition v we need constrain C_3 requiring that each pair of player plays at most once.

(d) Optionaly it would be good idea to create another constrain C_4 requiring order in variables of groups to prune state space.

One can see that although M_a model is quite straight forward for variable representation, we have to pay more attention for constrains.

2.1.1 Worst case estimate

Lets make estimate how much states CSP solver has to go through in worst case. Each variable V_i can be assigned by N domain values and there is $D * G * K$ of such variables thus estimate is:

$$N^{D*G*K} = 2^{\log_2(N)*D*G*K}$$

2.1.2 Conclusion

Problem of this model is obliviously unordered sets for group variables. This thought led us to following model.

2.2 Model dealing with unordered sets

This model increaces number of variables substantialy, but also reduces domains values substantialy.

Definition 2.2. Lets define CSP model $M_b = \langle V_b, D_b, C_b \rangle$ such as:

(i)

$$V_b = \{V_k | \forall i \in [D], \forall j \in [G], \forall l \in [N] \exists k \in \mathbb{N} : V_k \text{ is CSP variable } \wedge k = i*G*N+j*N+l\}$$

(ii)

$$D_b = \{D_i | \forall V_i \in V : D_i = [2]\}$$

(iii)

$$C_b = \{C_1, C_2, C_3\}$$

(a) To ensure condition ii we need constrain C_1 ensuring there is going to be set exactly K variables from each variables for given group.

(b) The condition iii is taken for granted in this model.

(c) To ensure condition iv we need constrain C_2 requiring that each player plays once per day.

(d) To ensure condition v we need constrain C_3 requiring that each pair of player plays at most once.

2.2.1 Worst case estimate

Lets make estimate how much states CSP solver has to go through in worst case. Each variable V_i can be assigned by 2 domain values and there is $D * G * N$ of such variables thus estimate is:

$$2^{D*G*N}.$$

2.2.2 Worst case estimate M_a vs M_b

Lets try to find out for, which N and G is M_b model better.

$$2^{D*G*N} < 2^{\log_2(N)*D*G*K}$$

$$D*G*N < \log_2(N)*D*G*K$$

$$N < \log_2(N)*K$$

Using remark from 1.1 $N = G*K$.

$$K*G < \log_2(N)*K$$

$$G < \log_2(N)$$

$$2^G < N$$

Which holds for N:

$$N \in \Omega(2^G)$$

Thus for $N >> G$ can be this model actually better.

2.2.3 Conclusion

To sum this up we have increased number of variables in favor of decreasing number of domain variables. We also need less constrains, because this model does not have problem with order of variables in groups. This model also automaticly ensures that one player can not be in group multiple times, which leads to simplification of constrains. Unfortunately condition $N \in \Omega(2^G)$ for better worst time estimate is quite hard.

Furthermore, there is still quite a few constrains which just ensure that players are assigned properly to groups. Following model thus try to look at problem from different perspective in order to simplify requried constrains.

2.3 Dual model

In this model there is met trade of between numbers of variables and numbers of values in domains. We are going to assign groups to players which reduces number of variables significantly.

Definition 2.3. Lets define CSP model $M_c = \langle V_c, D_c, C_c \rangle$ such as:

(i)

$$V_c = \{V_k | \forall i \in [D], \forall j \in [N] \exists k \in \mathbb{N} : V_k \text{ is CSP variable } \wedge k = i*N+j\}$$

(ii)

$$D_c = \{D_i | \forall V_i \in V : D_i = [G]\}$$

(iii)

$$C_c = \{C_1, C_2, C_3\}$$

(a) To ensure condition ii we need constrain C_1 ensuring that group assignment leds to groups of equal size K.

(b) The condition iii is taken for granted in this model, because each player in each day has assigned only one group.

(c) The condition iv is also taken for granted, because every player in each day has assigned group.

(d) We still need to ensure condition v with constrain C_2 requiring that each pair of player plays at most once.

(e) Optionaly we can add constrain C_3 reducing only solutions where days are accepted only in lexically ascending order - which reduces symetries.

2.3.1 Worst case estimate

For each day we assign groups to N players, which makes following estimate:

$$G^{D*N} = 2^{D*N*\log_2(G)}.$$

2.3.2 Worst case estimate M_a vs M_c

$$2^{D*N*\log_2(G)} < 2^{\log_2(N)*D*G*K}$$

$$D * N * \log_2(G) < \log_2(N) * D * G * K$$

$$N * \log_2(G) < \log_2(N) * G * K$$

Using remark from 1.1 $N = G * K$.

$$N * \log_2(G) < \log_2(N) * N$$

$$\log_2(G) < \log_2(N)$$

$$G < N$$

Which holds, thus M_c has better estimate than M_a.

2.3.3 Worst case estimate M_b vs M_c

$$2^{D*N*\log_2(G)} < 2^{D*G*N}$$

$$D * N * \log_2(G) < D * G * N$$

$$\log_2(G) < G$$

Which holds, thus M_c has better estimate than M_b.

2.3.4 Conclusion

This model gives best worst case estimate and has simpliest condition thus it is clear candidate for implementation, which is described in next section.

3 CSP model M_c implementation

All the variables are stored in *variables data structure* implemented as list of days, where each days has N variables determinig players group assignment.

Definition 3.1. *Variables data structure*(VDS) is matrix $\mathbb{D}$ with $D \times N$ dimensions such as:

$$\mathbb{D}_{i,j} = \begin{cases} \text{unbound} & \text{player } j \text{ in day } i \text{ does not have group assigned yet,} \\ g \in \{1, ..., G\} & \text{player } j \text{ in day } i \text{ has assgned group } g. \end{cases} \tag{1}$$

3.1 Group size constrain - `group_size_constrain`

This constrain should satisfy condition ii. For each day and each group constrain ensures that the same number of all assignment for each group is present for each day.

3.2 Pairs constrain – `pair_constrain`

This constrain should satisfy condition v. Because this condition basically binds every pair of variables we have decided to implement that as custom global constrain using The Global Constraint Programming Interface.

3.2.1 Constrain data structures

We set to wake up constrain every time arbitrary domain variable has changed. Once our predicate is called because of change we have to identify, which variables has actually changed. For that purpouse we define *pair constrain state* and *played record.*

Remark. In source code we define slang *variable* is *played*, which means *variable* has changed from *unbound* to *grounded variable.*

Definition 3.2. *Pair constrain state*(PCS) is matrix $\mathbb{S}$ with $D \times N$ dimensions (represented as list of lists) such as:

$$\mathbb{S}_{i,j} = \begin{cases} \text{ground} & \text{variable is grounded,} \\ \text{var} & \text{variable is unbound.} \end{cases} \tag{2}$$

Definition 3.3. *Played record*(PR) is matrix $\mathbb{P}$ with $D \times N$ dimensions (represented as list of lists) such as:

$$\mathbb{P}_{i,j} = \begin{cases} \text{ground} & \text{variable has been grounded,} \\ \text{none} & \text{variable did not changed,} \\ \text{unground} & \text{variable has became unbound .} \end{cases} \tag{3}$$

3.2.2 Psudo code algorithm

Upon wake up constrain process following steps.

1. Gets previous PCS $\mathbb{S}'$ and current VDS $\mathbb{D}$.

2. Calculates new PCS $\mathbb{S}$ from current VDS $\mathbb{D}$.
3. Based on $\mathbb{S}'$ and $\mathbb{S}$ creates PR $\mathbb{P}$.
4. Base on $\mathbb{P}$ for each grounded variable finds invalid domain values of each variables.
5. For each variable calculate new domains.
6. Unifies action with new domains and new PCS $\mathbb{S}$ with output parameters.

3.2.3 Invalid domain values detection

Lets assume we get notified that variable for n-th player in p-th day has been grounded.

Definition 3.4. Lets define following variables to simplify notation:

1. $dp := \mathbb{D}_{p,*}$ day with notified grounded variable,
2. $\forall p' \neq p : d := \mathbb{D}_{p',*}$ arbitrary other day,
3. $I_{i,j}$ set of invalid domain values for variable in i-th day of j-th player.

Following table illustrate which domains can be reduced by which type of reductions.

- o... denotes notified grounded variable
- A... denotes variable whose domain can be reduced by A reduction
- B... denotes variable whose domain can be reduced by B reduction
- C... denotes variable whose domain can be reduced by C reduction

	d_i	d_i	d_n	d_i	d_i	d_i
d	A	A	C	A	A	A
d	A	A	C	A	A	A
dp	B	B	o	B	B	B
d	A	A	C	A	A	A
d	A	A	C	A	A	A

Other day domains reduction – A Add value d_n to invalid set for i-th player in p'-th day. $\forall i \neq p \in [N]$ such as:

(i) d_n grounded

(ii) dp_i grounded

(iii) d_i not grounded variable whose domain can be reduced

(iv) $dp_n = dp_i$ n-th and i-th player are in the same group in p-th day

$$I_{p',i} \cup \{d_n\}$$

Played day domains reduction – B Add value dp_n to invalid set for i-th player in p-th day. $\forall i \neq p \in [N]$ such as:

(i) d_n grounded

(ii) dp_i not grounded variable whose domain can be reduced

(iii) d_i grounded

(iv) $d_n = p_i$ n-th and i-th player are in the same group in p'-th day

$$I_{p,i} \cup \{dp_n\}$$

Player domain reduction – C Add value d_i to invalid set for n-th player in p'-th day. $\forall i \neq p \in [N]$ such as:

(i) d_n not grounded variable whose domain can be reduced

(ii) dp_i grounded

(iii) d_i grounded

(iv) $dp_n = dp_i$ n-th and i-th player are in the same group in p-th day

$$I_{p',n} \cup \{d_i\}$$

Resulting algorithum thus for each notified grounded variable tries to apply all A, B, C reductions to populate sets of invalid domain values. Together with this step is also done check whether the grounded variables are valid. Using these invalid sets all possible domains are then reduced. Based on the result the fail action or actions changing domain sets are returned.

3.3 Days symetry constrain

To reduce day symetries we also provide solver variant, which searchs for only solutions with days assignment in lexically asecending order. To achieve this we use built-in constrain **`lex_chain`**.

4 User manual

Entire source code is present in file **`golf-problem-solver.pl`**. There are basically two predicates one can be interested in:

- **`golf`** - solves golf problem with symetries of exchanged days removed by defining asscending lexical order of days,
- **`golf_all`** - solves golf problem with all (even symetric) solutions.

There are following variants of these predicates with parameters description:

```
% golf_all(-DaysAttendance, +N, +K, +G, +D) :-
% golf(-DaysAttendance, +N, +K, +G, +D) :-
% golf_all(-T, -DaysAttendance, +N, +K, +G, +D, +Opt) :-
% golf(-T, -DaysAttendance, +N, +K, +G, +D, +Opt) :-
% T ... time of search in miliseconds
% N ... number of players
% K ... number of players in group
% G ... number of groups
% D ... number of days
% Opt ... labeling predicate options
```

Example call can be as follows:

```
| ?- golf(T, X, 20, 4, 5, 4, []).
T = 20920,
X = [[1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5],[1,2,3,4,1,2,3,5,1,2,4,5,1,3,4,5,2,3,4,5],[
yes

% formated X output
% 1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5
% 1,2,3,4,1,2,3,5,1,2,4,5,1,3,4,5,2,3,4,5
% 1,2,3,4,2,1,4,5,3,4,5,1,4,5,2,3,5,1,3,2
% 1,2,3,4,4,3,2,5,5,1,2,4,3,1,5,2,4,5,1,3
% ABCD | EFGH | IJKL | MNOP | QRST |
% AEIM | BFJQ | CGNR | DKOS | HLPT |
% AFLR | BEOT | CIPS | DGJM | HKNQ |
% AJNS | BGKP | CFMT | DELQ | HIOR |
```

To simplify running more jobs and getting better output we have created bash script **`test.sh`**. It automatically sets prologs print options and formates result output and even converts result to table of sets as it is in problem definition 1. Lastly it saves log output into file with name, metioning passed parameters, to log directory.

Script has following parameters:

```
test.sh <solver-variant> [N K G D <labeling-opts-params> <variants-count-to-search>]
<solver-variant> := basic|day_symetries
```

5 Tests

Beside some really simple variants such as in predicate **`test(X)`** we ran solver with parameters as in original problem 1.

$$N = 20, K = 4, G = 5, D = 5$$

Follows interesting part of logged solver output for variant which reduces symetries of days:

```
./logs/log-day_symetries-20-4-5-5-[]-10.txt

T = 4139770,
X = [[1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5],[1,2,3,4,1,2,3,5,1,2,4,5,1,3,4,5,2,3,4,5],[
1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5
1,2,3,4,1,2,3,5,1,2,4,5,1,3,4,5,2,3,4,5
1,2,3,4,2,1,4,5,3,4,5,2,4,5,1,3,5,2,3,1
1,2,3,4,3,4,5,2,4,1,3,5,2,4,5,1,5,1,2,3
```

```
1,2,3,4,4,5,2,3,2,3,5,1,5,1,3,4,4,5,1,2
ABCD | EFGH | IJKL | MNOP | QRST |
AEIM | BFJQ | CGNR | DKOS | HLPT |
AFOT | BELR | CIPS | DGJM | HKNQ |
AJPR | BHMS | CEKT | DFIN | GLOQ |
ALNS | BGIT | CHJO | DEPQ | FKMR |
T = 4139820,
X = [[1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5],[1,2,3,4,1,2,3,5,1,2,4,5,1,3,4,5,2,3,4,5],[
1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5
1,2,3,4,1,2,3,5,1,2,4,5,1,3,4,5,2,3,4,5
1,2,3,4,2,1,4,5,3,4,5,2,4,5,1,3,5,2,3,1
1,2,3,4,3,4,5,2,4,1,3,5,2,4,5,1,5,1,2,3
1,2,3,5,5,4,2,3,2,3,4,1,4,1,3,5,5,4,1,2
ABCD | EFGH | IJKL | MNOP | QRST |
AEIM | BFJQ | CGNR | DKOS | HLPT |
AFOT | BELR | CIPS | DGJM | HKNQ |
AJPR | BHMS | CEKT | DFIN | GLOQ |
ALNS | BGIT | CHJO | FKMR | DEPQ |
```

Another par of logged solver output, which is not reducing symetries, is as follows:

```
./logs/log-basic-20-4-5-5-[]-10.txt

T = 7714440,
X = [[1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5],[1,2,3,4,1,2,3,5,1,2,4,5,1,3,4,5,2,3,4,5],[
1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5
1,2,3,4,1,2,3,5,1,2,4,5,1,3,4,5,2,3,4,5
1,2,3,4,2,1,4,5,3,4,5,2,4,5,1,3,5,2,3,1
1,2,3,4,3,4,5,2,4,1,3,5,2,4,5,1,5,1,2,3
1,2,3,4,4,5,2,3,2,3,5,1,5,1,3,4,4,5,1,2
ABCD | EFGH | IJKL | MNOP | QRST |
AEIM | BFJQ | CGNR | DKOS | HLPT |
AFOT | BELR | CIPS | DGJM | HKNQ |
AJPR | BHMS | CEKT | DFIN | GLOQ |
ALNS | BGIT | CHJO | DEPQ | FKMR |
T = 7714500,
X = [[1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5],[1,2,3,4,1,2,3,5,1,2,4,5,1,3,4,5,2,3,4,5],[
1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5
1,2,3,4,1,2,3,5,1,2,4,5,1,3,4,5,2,3,4,5
1,2,3,4,2,1,4,5,3,4,5,2,4,5,1,3,5,2,3,1
1,2,3,4,3,4,5,2,4,1,3,5,2,4,5,1,5,1,2,3
1,2,3,5,5,4,2,3,2,3,4,1,4,1,3,5,5,4,1,2
ABCD | EFGH | IJKL | MNOP | QRST |
AEIM | BFJQ | CGNR | DKOS | HLPT |
AFOT | BELR | CIPS | DGJM | HKNQ |
AJPR | BHMS | CEKT | DFIN | GLOQ |
ALNS | BGIT | CHJO | FKMR | DEPQ |
```

We can see that reducing symetries is really usefull, because compute time is nearly 2 times faster. However for smaller problems the difference is not that interesting.

We have also tried different labeling parametr options, however default behaviour appeared to be the fastest one. You can found all the logs in `logs` folder to see the difference in durations.