iZ-C

A Constraint Programming Library

Toshimitsu FUJIWARA
NTT DATA SEKISUI SYSTEMS CORPORATION

16 Sep, 2013

NTT DATA
株式会社NTTデータセキスイシステムズ

About us

- NTT DATA SEKISUI SYSTEMS CORPORATION
  - Business
    - Application development
    - IT infrastructure building and management
    - And related services
    - （"iZ-C" is one of software tools developed for our business)
  - http://www.ndis.co.jp/

- About me
  - Toshimitsu FUJIWARA
  - Software developer
  - Currently maintaining iZ-C (2009-)
  - Author of "izplus" (FlatZinc Solver)

Presentation Overview

- Introduction
  - What is iZ-C?
- Inside iZ-C
- Applications in Real world
  - Shift-kun
  - Film cutting planning
  - Train driver rostering
  - izplus
- Summary and Conclusion

What is iZ-C?

- Library written in C
- Finite domain constraint solver
- Practical constraints
  - Arithmetic constraints
  - High level global constraints
  - Reifications
- Extensible
  - User can write own constraint and search mechanism.

SEND + MORE = MONEY

```
#include <stdio.h>

#include "iz.h"

CSint **Digit;
CSint *L1, *L2, *L3;

enum {s = 0, e, n, d, m, o, r, y, NB_DIGITS };

void constraints ()
{
    Digit = cs_createCSintArray (NB_DIGITS, 0, 9);

    L1 = cs_VScalProd (4, Digit [s], Digit [e], Digit [n], Digit [d],
       1000, 100, 10, 1);

    L2 = cs_VScalProd (4, Digit [m], Digit [o], Digit [r], Digit [e],
       1000, 100, 10, 1);

    L3 = cs_VScalProd (5, Digit [m], Digit [o], Digit [n], Digit [e], Digit [y],
       10000, 1000, 100, 10, 1);

    cs_Eq (L3, cs_Add (L1, L2));

    cs_NEQ (Digit [s], 0);
    cs_NEQ (Digit [m], 0);
    cs_AllNeq (Digit, NB_DIGITS);
}
```

```
void printSolution ()
{
    cs_printf (" %D¥n", L1);
    cs_printf ("+%D¥n", L2);
    cs_printf ("-----¥n");
    cs_printf ("%D¥n", L3);
    cs_printStats ();
}

int main (int argc, char **argv)
{
    cs_init ();

    constraints ();

    if  (cs_search (Digit, NB_DIGITS, cs_findFreeVarNbElements))
         printSolution ();
    else
         printf ("fail!¥n");

    cs_end ();
    return 0;
}
```

Inside iZ-C

- Introduction
  - What is iZ-C?
- Inside iZ-C
- Applications in Real world
  - Shift-kun
  - Film cutting planning
  - Train driver rostering
  - izplus （FlatZinc Solver)
- Summary and Conclusion

Mechanisms

- Integer domains represented using bitmap
- Pooling for high performance memory management
- Efficient codes for constraint propagation
  (Library user can create new constraint using callback.)

Extensive search

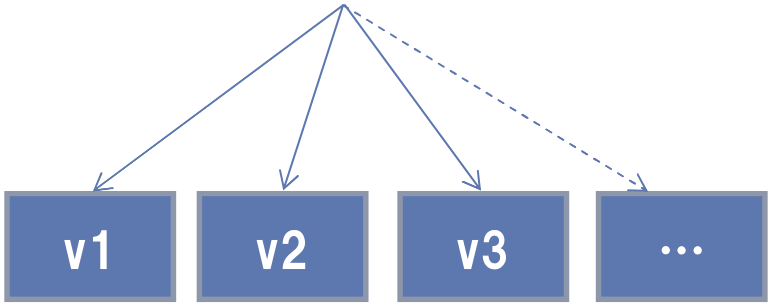
- User defined variable order to assign value
- User defined value order to assign to variable
- Save/Restore context for user defined search

Default search mechanism is not so powerful in comparison to modern solvers, but…

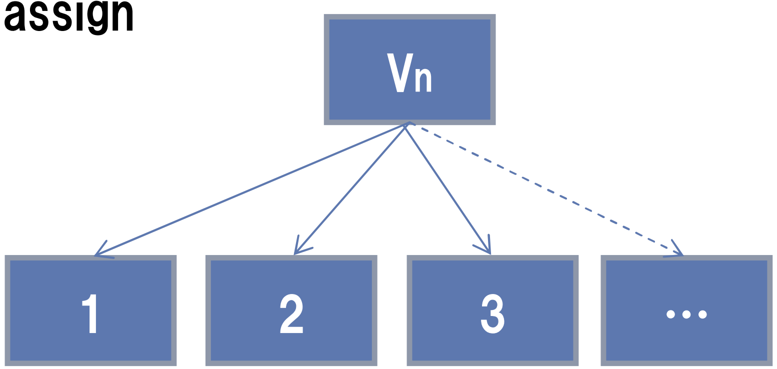
User control in search

```
IZBOOL cs_searchCriteriaFail(CSint **allvars, int nbVars,
                int (*findFreeVar)(CSint **allvars, int nbVars),
                int (*criteria)(int index, int val),
                int NbFailsMax)
```

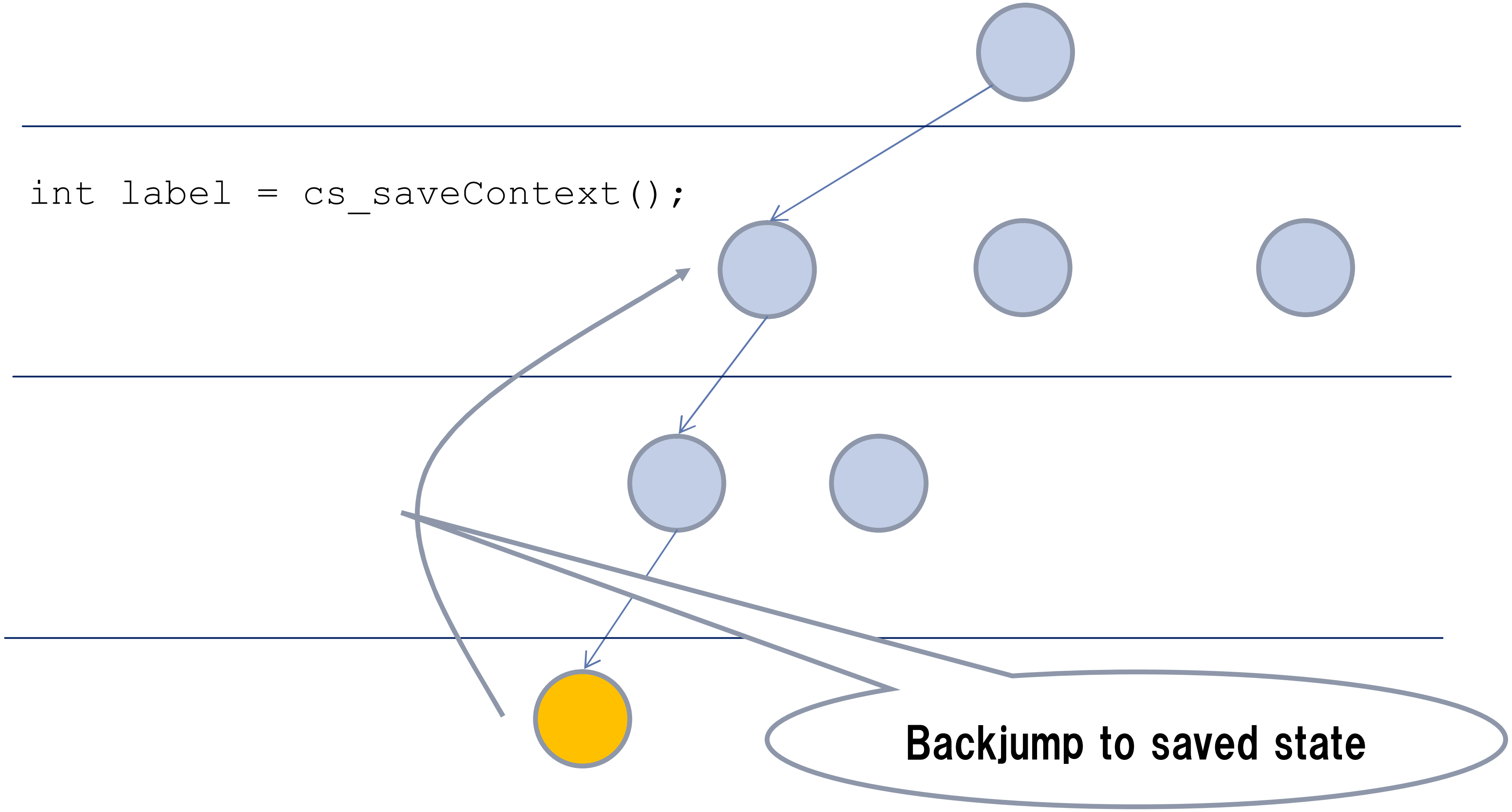
Controls variable selection order to instantiate



Controls value selection order to assign



Save/Restore context



```
int label = cs_saveContext();
```

Backjump to saved state

```
cs_restoreContextUntil(label);
```

C# Interface

- Developed for in-house use.
  - Can write own constraint in C# with modern features
    - Various predefined classes
    - Reflection, lambda expression, …
    - Garbage collection
    - IDE aided development and debugging
  - Easy to add GUI
    - GUI is as important as performance of solver in real application.

Applications in Real world

- Introduction
  - What is iZ-C?
- Inside iZ-C
- Applications in Real world
  - Shift-kun
  - Film cutting planning
  - Train driver rostering
  - izplus （FlatZinc Solver)
- Summary and Conclusion

Shift-kun

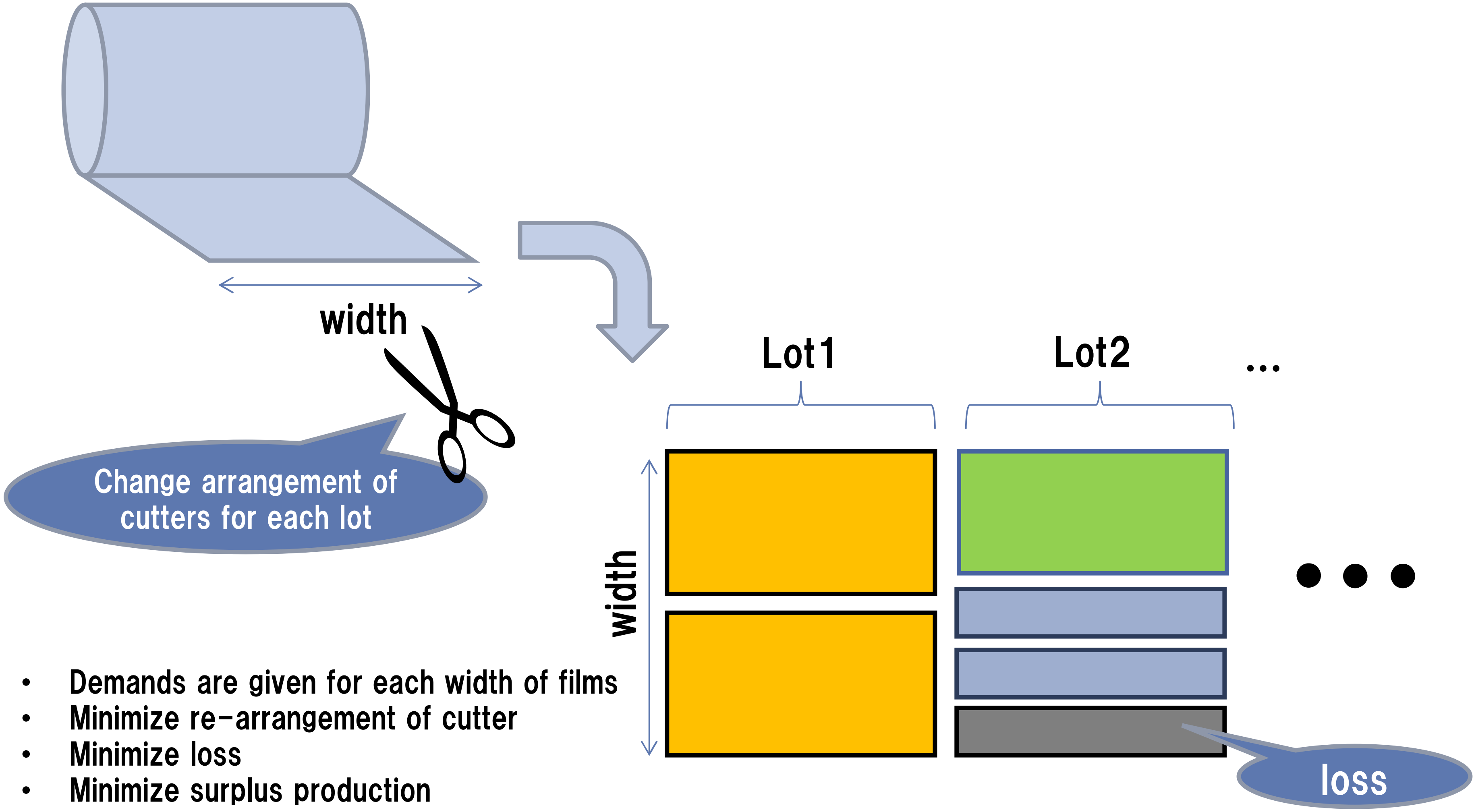
- One of best selling staff rostering application

Staff has his/her own preference

	day1	day2	day3	day4	day5	…
Staff1	Day	Night	Next day of Night Shift	off	Night	
Staff2	off	Day	Day	Day	Day	
Staff3	Day	off	Night	Next day of Night Shift	off	
Staff4	Night	Next day of Night Shift	off	Day	Day	
Staff5	off	Day	Day	Night	Next day of Night Shift	

＊ Satisfy demands of each facilities for every day
＊ Staff pair（prohibited or forced)

Film cutting planning



- Demands are given for each width of films
- Minimize re-arrangement of cutter
- Minimize loss
- Minimize surplus production

Train driver rostering（1）

- Train driver scheduling problem consists of two phases.
  - Daily operation schedule

    Multi depot VRP
    (CP can be used, but hard)

    - Starts from base station, operate trains station to station, and finally returns to base station.（in 1 day or 2day）
  - Roster for each driver

    Natural to use CP

    - To cover each daily operation, each driver starts different day in one schedule.（see next figure）

Train driver rostering（2)

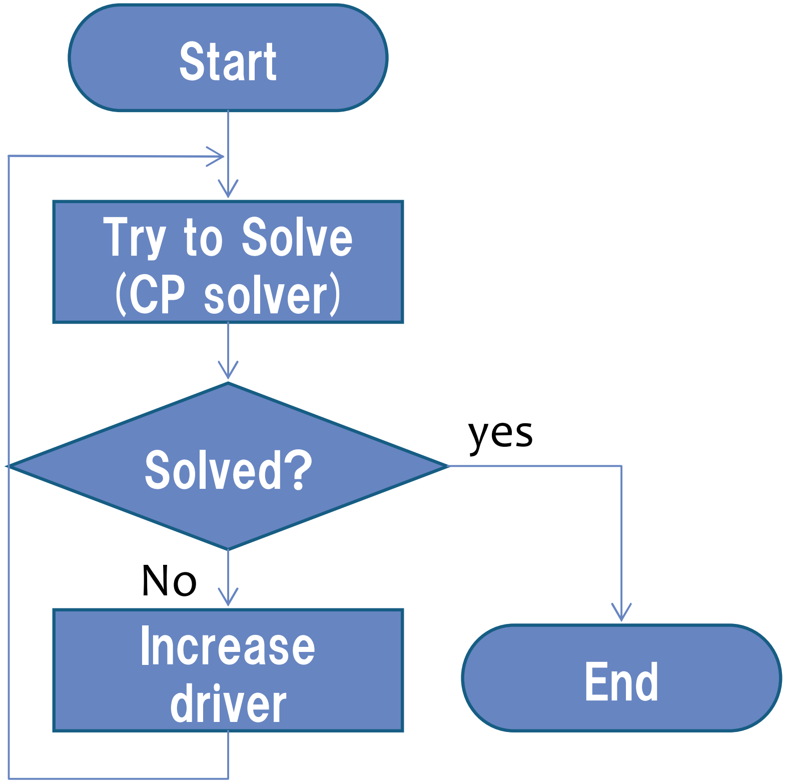
	day1	day2	day3	day4	day5	
Base plan	op1	op2	op3	off	op4	…
Driver1	op1	op2	op3	off	op4	…
Driver2	op2	op3	off	op4	op1	…
Driver3	op3	off	op4	op1	op2	…
Driver4	off	op4	op1	op2	op3	…
Driver5	op4	op1	op2	op3	off	…

Each operation is covered by driver for every day.

Train driver rostering（3）

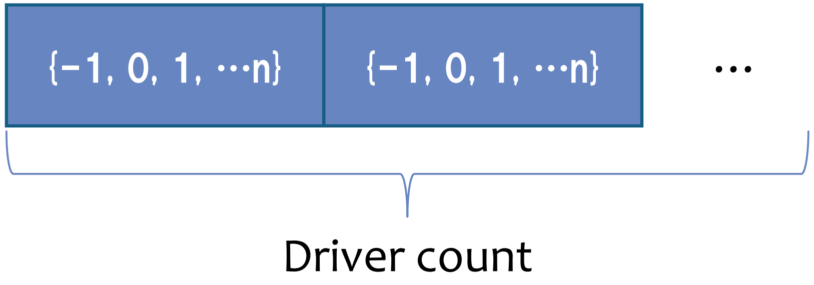
- Minimize needed driver count.
- Constraints
  - Must contain all daily operations.
  - Must contain predetermined day off per day count.
  - Prohibited arrangement patterns for operation's attribute.（ex. Continued night operation)

Train driver rostering (4)



{−1, 0, 1, …n}

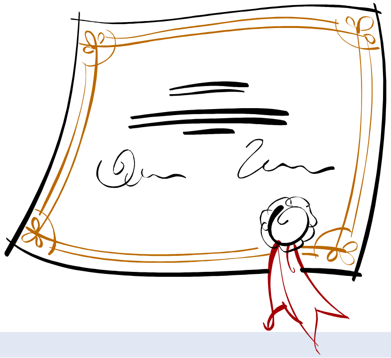
- -1 Day off
- 0 Place holder (2nd day of 2 day length operation)
- 1..n operation-n



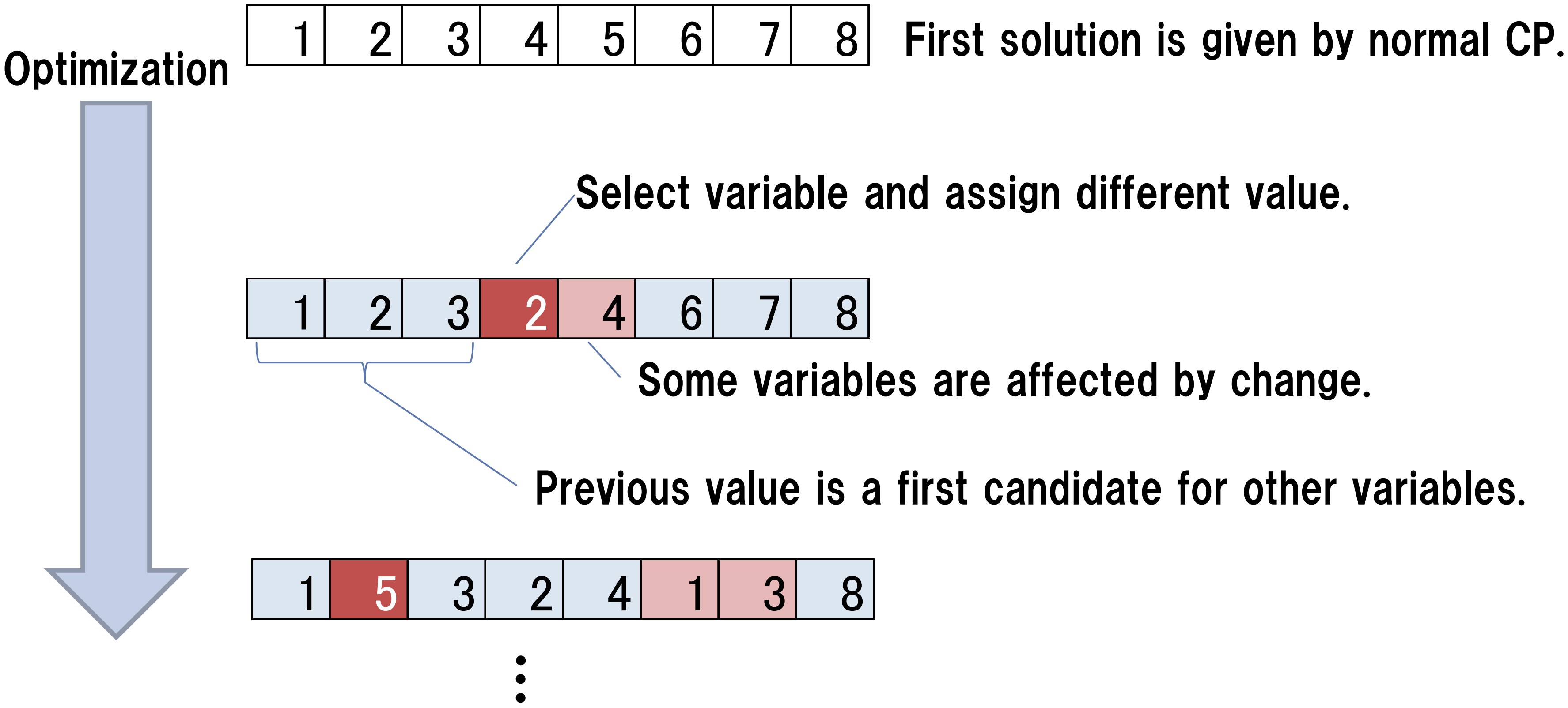
- Each value in {1..n} must appear just one time.
- Appearance count of -1 is determined by driver count.
- 0 is assigned if any other value cannot be assigned.

izplus（FlatZinc Solver)

- FlatZinc solver developed using iZ-C
    - Extended using “Random restart” and “Local search”
- Participant of MiniZinc Challenge 2012
- Bronze medal in two categories
    - Free search
    - Parallel search



Local search with iZ-C (1)



Optimization

1	2	3	4	5	6	7	8

First solution is given by normal CP.

Select variable and assign different value.

1	2	3	2	4	6	7	8

Some variables are affected by change.

Previous value is a first candidate for other variables.

1	5	3	2	4	1	3	8

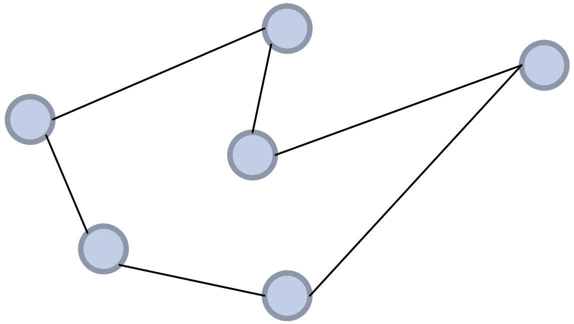
⋮

Same procedure is repeated for better objective value.

Local search with iZ-C（2）

- Pitfall of CP based search
  - Sometimes constraint propagation cannot delete enough candidate values from domain variables.
  - In such case, search fails very late phase of explore.

- Advantage of local search
  - Good solutions are similar to each other.
  - Can preserve partial structure of solution.



Summary and Conclusion

- Introduction
  - What is iZ-C?
- Applications in Real world
  - Shift-kun
  - Film cutting planning
  - Train driver rostering
  - izplus（FlatZinc Solver)
- Summary and Conclusion

Summary and Conclusion

- iZ-C is a library for constraint programming.
  - Efficient and extensive
  - By integrating to modern language, we can use high level functions and library including GUI.

- Applied to many problems but..
  - CP is powerful but not enough.
    We need deep insights of particular problems to create good heuristics for variable/value ordering.
    Sometimes we need CP to be combined with other methods（ex. local search）.

- iZ-C can satisfy such needs.
  - Of course, we need more research and development!

Appendix

- More information about iZ-C（written in Japanese)
  - http://solution.ndis.jp/iz/

- MiniZinc Challenge 2012 Results
  - http://www.minizinc.org/challenge2012/results2012.html