A General Purpose Local Search Solver

October 6, 2015

Contents

1 Introduction

1.1 Mixed Integer Programming

1.2 Constraint Programming

1.3 Heuristics and Local Search

1.3.1 Construction Heuristics

1.3.2 Local Search and Neighborhoods

1.3.3 Metaheuristics

2 Modeling

2.1 Variables and Constraints

Models contains variables X that is a n-tuple of variables $X = \langle x_1, x_2, \ldots, x_n \rangle$. Each variable $x_i \in X$ has a domain $D_i \in D$ where D is an n-tuple of domains $D = \langle D_1, D_2, \ldots, D_n \rangle$ such that $x_i \in D_i$. The variables $x_i \in X$ of the models that will be discussed in this paper all have the domain restricted to $D_i \subseteq \mathbb{Z} \ : \forall \ D_i \in D$ **(Does not look quite nice and is not true for MIP)**. The variables will be restricted by C that is a m-tuple of constrains $C = \langle C_1, C_2, \ldots, C_m \rangle$. The set of variables to which the constraint C_j applies is called its scope and is denoted S_j. Each $C_j \in C$ is a pair $\langle R_{S_j}, S_j \rangle$ where R_{S_j} is a subset of the cartesian product of the domains of the variables in S_j also called the relation on C_j.
The constraint satisfaction problem (CSP) can then be defined as a triple $\mathbb{P} = \langle X, D, C \rangle$. A solution to the CSP P is an n-tuple $A = \langle a_1, a_2, \ldots, a_n \rangle$ where $a_i \in D_i$. The solution is feasible if the projection of A onto S_j is included in R_{S_j} for all $C_j \in C$.
The solution of interest could be all feasible solutions $sol(P)$, any feasible solution A or if there exists a solution or not.

2.2 Types of Modeling

The choice of modeling approach should depend on the nature of the problem. Constraint programming (CP) uses constraints to formulate a constraint satisfaction problem (CSP) and often provides short and natural way to describe a problem. Constraint programming try to find feasible solutions to a CSP hence it does not differentiate the solutions that might exist. One can work around that by creating a function that evaluates a solution by a value. Then add the constraint that the function must give a better value than last time.
(Jeg ved ikke om jeg skal definere IP her eller tidligere (og hvor meget der skal skrive))
Mixed integer programming (MIP) and Integer programming (IP) are more restricted in the choice of constraints. They can only use linear constraints hence the models often tend to be longer in order to formulate a problem. Unlike constraint programming both mixed integer programming and integer programming has an objective function that rates the quality of a solution. Binary programming (BP) models the same way as MIP and IP but restricts the variables from

integer to binary.

3 Different Solvers (properly not the best name)

3.1 Comet

Invariant bliver introduceret her, så endten skal det være efter min struktur eller også skal jeg definere invariant og oneway før det her.

3.2 Gecode

3.3 LocalSolver

3.4 OscaR

3.5 ((This solver)) Constraint Based Local Search with Limitations

4 Preprocessing and Simplification

4.1 Gecode Engine

4.1.1 Relaxation

4.2 Initial Solution

5 Structuring Local Search Model

5.1 Simplification

5.2 Invariants and One-way Constraints

5.3 Maintaining Invariants

5.4 Maintenance Graph

6 Local Search Engine

6.1 Neighborhoods

6.1.1 Moves

6.2 Metaheuristics

7 Tests

8 Results

9 Conclusion

```
special treatment of the first line;
for i ← 2 to l do
    special treatment of the first element of line i;
    for j ← 2 to w do
        left ← FindCompress(Im[i, j − 1]);
        up ← FindCompress(Im[i − 1, ]);
        this ← FindCompress(Im[i, j]);
        if left compatible with this then // O(left,this)==1
            if left < this then Union(left,this);
            ;
            else Union(this,left);
            ;
        end
        if up compatible with this then              // O(up,this)==1
            if up < this then Union(up,this);
            ;
            // this is put under up to keep tree as flat as
               possible
            else Union(this,up);
            ;                                  // this linked to up
        end
    end
    foreach element e of the line i do FindCompress(p);
end
```

Algorithm 1: disjoint decomposition