Test: Comparison with standard CP

F. Campeotto[1,2], A. Dovier[1], F. Fioretto[1,2], and E. Pontelli[2]

[1] Dept. Mathematics & Computer Science, Univ. of Udine
[2] Dept. Computer Science, New Mexico State Univ.

1 Standard CP

We evaluate the performance of the GPU-LNS solver on some Minizinc benchmarks, comparing its results against the solutions found by the state-of-the-art CP solver JaCoP. We present results on medium-size problems which are neither too hard to be solved with standard CP techniques nor too small to make a local search strategy useless.

We considered the following four problems:[3] (1) The *Trasportation* problem, with only 12 variables but the optimal solution is hard to find using CP. The heuristics used for JaCoP is the `first_fail`, `indomain_min`, while for GPU-LNS we used the RL method.
We used $t = 100$ neighborhoods of size 3, $m = 100$ SP each, and $h = 500$. (2) The *TSP* with 240 cities and some flow constraints. The heuristics used for JaCoP is the same as above, RP strategy is used in GPU-LNS with $p = 1$. We use $t = 100$ neighborhood of size 40, $m = 100$, and $h = 5000$. (3) The *Knapsack* problem. We considered instances of 100 items[4]. The strategy adopted in JaCoP is `input_order`, `indomain_random`, while for G-LNS we used the RL search strategy, with $t = 50$ neighborhoods of 20 variables, $m = 50$, and $h = 5000$. (4) The *Coins_grid* problem. We considered this problem to test our solver on a *highly* constrained problem. For this benchmark we sightly modified the LS strategy: we used CP to generate random starting points without performing any labeling in order to preserve the initial random solution found fpr each large set of variables. The strategy adopted in JaCoP is `most_constrained`, `indomain_max`, while for G-LNS we used the RL search strategy, with $t = 300$ neighborhoods of 20 variables, $m = 150$, and $h = 50000$. Table 1 reports the first solution value, the best solution found (within 10 min) and the (average on 20 runs for GPU-LNS) running times. For GPU-LNS we also report the standard deviation of the best solution value.

[3] Models and description are available at http://www.hakank.org/minizinc/
[4] An hard instance has been generated using the generator that can be found at `http://www.diku.dk/~pisinger/generator`.

Table 1: Minizinc benchmarks (minimizazion problems, save Knapsack).

System	Benchmark	First Sol	Best Sol(sd)	Time(s)
JaCoP	Transportation	6699	6640	600
JaCoP	TSP	10098	6307	600
JaCoP	Knapsack	7366	15547	600
JaCoP	Coins_grid	20302	19478	600
GPU-LNS	Transporation	7600	5332 (56)	57.89
GPU-LNS	TSP	13078	6140 (423)	206.7
GPU-LNS	Knapsack	0	48219 (82)	6.353
GPU-LNS	Coins_grid	20302	16910 (0)	600