INFO-Y087: Declarative Programming
Project: Task Scheduler Problem

Brian Delhaisse

August 21, 2015

The goal of this project was to solve the Task Scheduler Problem on multiple instances using the Prolog programming languages. They were 10 instances divided in two groups: the small instances and the large ones. In this new version of the project, all the required predicates work for all the instances.

Here is the list of updates done in this version:

- `find_heuristically(-S)`: This predicate has been implemented for the large instance. For now, this predicate calls another one `find_randomly(-Sol, +NbIteration, -Cost)`. The predicates that have been implemented when designing the heuristic are:
  - `find_randomly(-Sol, +NbIteration, -Cost)` which creates NbIteration of random solutions and keeps the best one (that is, the one which is minimizing the cost).
  - `buildGreedy(-Sol, -Cost)` which constructs incrementally the solution by minimizing the direct cost.
  - `find_subOptimal(-Sol, +size)` which for now only works when there are no dependencies. It divides the list of tasks into multiple lists of the specified size, apply the `find_optimal` on those small lists, then merge the results. If the size is equal to the number of tasks this will give the same result than `find_optimal`.
- Many other useful predicates have been implemented such as `merge(+S1,+S2,-S)` which merges two solutions into one, `saveSolution(+Filename, +Solution, +Cost, +Method)` which saves the solution in a file, `assign(+Task, +Core, +InitSol, -FinalSol)` which assigns a task to a core in the solution given in argument, `predCore/predTask(-Delta, +V1, +V2)` which are used by `predsort/3` to sort the list of tasks or cores[1], etc.
- `isSolution(+S)`, `speedup(+S,-Speedup)`, `find_optimal(-S)`, `pretty_print(+S)`, `and execution_time(+S,-ET)` have been unchanged since the previous version.

All the predicates should work for all the instances. I obtained the same results than the results written on the project webpage[2] for the small instances.

In the following table, the reader can find the cost and speedup found for the large instances:

[1] We need this because of an assumption that I made: I assume that the tasks are written in the order of dependency, that is, they are written like `depends_on(Tn,Tm,_)` with $n > m$.

[2] http://ai.vub.ac.be/node/1353

Name	method	exec. time	speedup	time(sec)
batch_large_homo	random(1000) random(10000)	1169	13.77	6.91
	buildGreedy	16102	1	0.06
	subOptimal(1)	16102	1	0.52
	subOptimal(2)	8499	1.895	3.92
	subOptimal(3)	6794	2.370	42.34
	subOptimal(4)	5445	2.957	538
batch_large_hetero	random(1000)	1152	13.27	7.06
	buildGreedy	177	86.384	0.10
	subOptimal(1)	177	86.384	0.50
	subOptimal(2)	177	86.384	4.15
	subOptimal(3)	208	73.510	44.23
	subOptimal(4)	257	59.494	533
fib_large_nc	random(1000)	1253	2.116	22.86
	buildGreedy	2650	1	0.08
fib_large_uc	random(1000)	2488	1.065	23.09
	buildGreedy	2650	1	0.09
sor_large	random(1000)	6612	1.282	24.68
	buildGreedy	8480	1	0.05

About the time, it has been measured using the predicate `time/1`, and has been averaged over 3 runs except for subOptimal(4). All the instances and the code have been run on a MacbookPro 13 (version OS 10.9.4), Intel Core i7, 2.9GHz, 8GB 1600MHz DDR3. Concerning the cost for the "random" method, it has also been averaged over 3 runs. The same goes for the speed up.

In order to use the code, the user needs to load the project first then the instances using the predicate `load(+NameOfInstance)`. For large instances, the reader will probably need to set a flag to display the whole result `set_prolog_flag(toplevel_print_options, [quoted(true), portray(true), max_depth(400), spacing(next_argument)])`.

In the code, the call to the predicate `saveSolution/4` has been commented, so no solutions will be saved in a file.

Appendix

Here are the examples that were in the previous version of the project:

```
?- [project].
% project compiled 0.00 sec, 71 clauses
true.

?- load(batch_small_homo).
% batch_small_homo compiled 0.00 sec, 21 clauses
true.

?- getSolution(S).
S = solution([schedule(c1, [t1, t2, t3, t4, t5, t6|...]), schedule(c2, []),
    schedule(c3, []), schedule(c4, [])]) ;
S = solution([schedule(c1, [t1, t2, t3, t4, t5, t6]), schedule(c2, [t7]),
    schedule(c3, []), schedule(c4, [])]) ;
S = solution([schedule(c1, [t1, t2, t3, t4, t5, t6]), schedule(c2, []), schedule(
    c3, [t7]), schedule(c4, [])]) ;
S = solution([schedule(c1, [t1, t2, t3, t4, t5, t6]), schedule(c2, []), schedule(
    c3, []), schedule(c4, [t7])]) ;
S = solution([schedule(c1, [t1, t2, t3, t4, t5, t7]), schedule(c2, [t6]),
    schedule(c3, []), schedule(c4, [])]) ;
S = solution([schedule(c1, [t1, t2, t3, t4, t5, t7]), schedule(c2, []), schedule(
    c3, [t6]), schedule(c4, [])]) ;
S = solution([schedule(c1, [t1, t2, t3, t4, t5, t7]), schedule(c2, []), schedule(
    c3, []), schedule(c4, [t6])]) ;
S = solution([schedule(c1, [t1, t2, t3, t4, t5]), schedule(c2, [t6, t7]),
    schedule(c3, []), schedule(c4, [])]) ;
S = solution([schedule(c1, [t1, t2, t3, t4, t5]), schedule(c2, [t6]), schedule(c3
    , [t7]), schedule(c4, [])]) ;
...

?- find_optimal(S).
S = solution([schedule(c1, [t1]), schedule(c2, [t2, t7]), schedule(c3, [t3, t6]),
     schedule(c4, [t4, t5])]).

?- execution_time(solution([schedule(c1, [t1]), schedule(c2, [t2, t7]), schedule(
    c3, [t3, t6]), schedule(c4, [t4, t5])]), ET).
ET = 100.

?- speedup(solution([schedule(c1, [t1]), schedule(c2, [t2, t7]), schedule(c3, [t3
    , t6]), schedule(c4, [t4, t5])]),SP).
SP = 4.

?- pretty_print(solution([schedule(c1, [t1]), schedule(c2, [t2, t7]), schedule(c3
    , [t3, t6]), schedule(c4, [t4, t5])])).
The solution is:
On the core c1 : [t1]
On the core c2 : [t2,t7]
On the core c3 : [t3,t6]
On the core c4 : [t4,t5]
Total Execution Time: 100 ms.
true.

?- unload(batch_small_homo).
true.

?- load(batch_small_hetero).
```

```
% batch_small_hetero compiled 0.00 sec, 35 clauses
true.

?- find_optimal(S).
S = solution([schedule(c1, [t3, t7]), schedule(c2, [t1]), schedule(c3, [t4, t5]),
     schedule(c4, [t2, t6])]).

?- execution_time(solution([schedule(c1, [t3, t7]), schedule(c2, [t1]), schedule(
    c3, [t4, t5]), schedule(c4, [t2, t6])]), ET).
ET = 100.

?- speedup(solution([schedule(c1, [t3, t7]), schedule(c2, [t1]), schedule(c3, [t4
    , t5]), schedule(c4, [t2, t6])]), SP).
SP = 3.6.

?- unload(batch_small_hetero).
true.

?- load(fib_small_nc).
% fib_small_nc compiled 0.00 sec, 22 clauses
true.

?- find_optimal(S).
S = solution([schedule(c1, [t1, t2, t4, t6, t7]), schedule(c2, [t3, t5]),
    schedule(c3, []), schedule(c4, [])]).

?- execution_time(solution([schedule(c1, [t1, t2, t4, t6, t7]), schedule(c2, [t3,
     t5]), schedule(c3, []), schedule(c4, [])]), ET).
ET = 50.

?- speedup(solution([schedule(c1, [t1, t2, t4, t6, t7]), schedule(c2, [t3, t5]),
    schedule(c3, []), schedule(c4, [])]), SP).
SP = 1.4.

?- unload(fib_small_nc).
true.

?- load(fib_small_uc).
% fib_small_uc compiled 0.00 sec, 23 clauses
true.

?- find_optimal(S).
S = solution([schedule(c1, [t1, t3, t4, t5, t6, t7]), schedule(c2, [t2]),
    schedule(c3, []), schedule(c4, [])]).

?- execution_time(solution([schedule(c1, [t1, t3, t4, t5, t6, t7]), schedule(c2,
    [t2]), schedule(c3, []), schedule(c4, [])]), ET).
ET = 60.

?- speedup(solution([schedule(c1, [t1, t3, t4, t5, t6, t7]), schedule(c2, [t2]),
    schedule(c3, []), schedule(c4, [])]), SP).
SP = 1.1666666666666667.

?- isSolution(solution([schedule(c1, [t1, t3, t4, t5, t6, t7]), schedule(c2, [t2
    ]), schedule(c3, []), schedule(c4, [])])).
true.

?- isSolution(solution([schedule(c1, [t1, t3, t4, t5, t6, t7]), schedule(c2, [t2
    ]), schedule(c3, []), schedule(c4, [t20])])).
false.
```

```

?- isSolution(solution([schedule(c1, [t1, t3, t4, t5, t6, t7]), schedule(c2, []),
     schedule(c3, []), schedule(c4, [])])).
false.

?- isSolution(solution([schedule(c1, [t7, t3, t4, t5, t6, t1]), schedule(c2, [t2
    ]), schedule(c3, []), schedule(c4, [])])).
false.

?- unload(fib_small_uc).
true.

?- load(sor_small).
% sor_small compiled 0.00 sec, 51 clauses
true.

?- find_optimal(S).
S = solution([schedule(c1, [t1, t2]), schedule(c2, [t3, t6]), schedule(c3, [t4]),
     schedule(c4, [t5])]).

?- execution_time(solution([schedule(c1, [t1, t2]), schedule(c2, [t3, t6]),
    schedule(c3, [t4]), schedule(c4, [t5])]), ET).
ET = 174.

?- speedup(solution([schedule(c1, [t1, t2]), schedule(c2, [t3, t6]), schedule(c3,
     [t4]), schedule(c4, [t5])]), SP).
SP = 1.4252873563218391.

?- unload(sor_small).
true.
```