Advanced Programming Languages for AI
Task 3

Karl Dox Li Quan Nathan Vandecauter

July 1, 2011

Abstract

This is the report for the (optional) task 3. We will briefly discuss both parts of this task. First, we give an overview of possible interesting topics with respect to A.I. Second, we give a concrete example of the power of constraint programming by discussing the Survo puzzle in ECLiPSe.

Part I
Task 3

1 Introduction

This paper briefly discusses some aspects of Artificial Intelligence which could be interesting for a course 'Advanced Programming Languages for AI'.

Note that the selection of topics might be biased as all of them are based on tasks given in the course 'Capita Selecta AI' of the year 2010–2011.

2 Stochastic/probabilistic programming

> Although many ways have been proposed to model uncertain quantities, stochastic models have proved their flexibility and usefulness in diverse areas of science. This is mainly due to solid mathematical foundations and theoretical richness of the theory of probability and stochastic processes, and to sound statistical techniques of using real data. [17]

Optimization

For instance, consider the Traveling Tournament Problem [5, 18]. This is a very hard problem (even for small problem instances) as it combines two

problems [2]: first, the generation of schedules, and second, the optimization of those schedules. While for instance constraint programming techniques have been successfully used for the former, alone they do not suffice to incorporate the latter aspect [19].

At first integer programming techniques [6] were used for the optimization aspect, resulting in a hybrid algorithm. However, soon stochastic programming techniques such as simulated annealing [9] have been succesfully used which performed significantly better [16, 1, 21]. To date, these kind of search algorithms achieve (one of the) best results for many problem instances on [18].

Search

Another example would be searching embeddings in random graphs. Searching an exact number of a certain pattern on massive graphs is computationally infeasible; randomized algorithms also are reported to leverage these problems. [7, 14].

Modeling/Inference

The above given examples focus more on stochastic optimization and search which is a very interesting subfield of artificial intelligence/optimization in itself.

Of course, there also have been probabilistic programming languages developed, e.g. Problog [8], which are more focused on modeling the uncertainties of the world. This allows the user to make inductive inference which have a probability associated: an interesting application is in the area of robotics [10] and multi-agent systems. Also extensions of Problog such as DTProblog [20] (and the closely related Markov Decision Processes) and CP-Logic [22] can be studied.

3 Programming languages

It is important to note that Artificial Intelligence is a very broad topic. A selection of programming languages is thus always very difficult.[1] One of the most important things to consider is the need to combine different strategies (and thus also programming languages) [3, 13] based on the task (domain) at hand.

[1] For an interesting discussion on 'good' programming languages for AI, see for instance `http://stackoverflow.com/questions/82036/what-is-a-good-programming-language-for-ai`.

4 Constraint programming in imperative languages

While for instance ECLiPSe integrates constraints in a logic programming language (Prolog) in a "natural" way, sometimes—mostly for performance issues—users want to benefit from the declarative paradigm using imperative languages such as C++ (e.g. Gecode) [4]. A specific example would be using constraint programming for pattern mining [15].

5 Conclusion

It is clear that selecting a subset of good topics is not easy. While most of the topics discussed and also given in the course are mostly "academical" topics — which are interesting on their own, it would also be refreshing to actually use some AI programming language coupled with a more "real-world" project. However, this would also require more effort and it might not be so easy to organize.

Part II
Task 4

As an example of the power of constraint programming, we consider the Survo puzzle [23].

6 Introduction

This introduction is taken from [11].

> A class of cross sum puzzles called Survo puzzles will be studied. Survo puzzles are somewhat related to Kakuro cross sum puzzles but played on a simple, open rectangular grid and without limiting to integers from 1 to 9.
>
> In a Survo puzzle the task is to fill an $m \times n$-table by integers $1, 2, \ldots, mn$ so that each of these numbers appears only once and their row and column sums are equal to integers given on the bottom and the right side of the table. Often some of the integers are given readily in the table in order to guarantee uniqueness of the solution and/or for making the task easier.

As an illustration, consider the following example and its solution:

	6			30
				18
		3		30
27	16	10	25	

12	6	2	10	30
8	1	5	4	18
7	9	3	11	30
27	16	10	25	

7 Constraint programming

It should be clear that this problem is very easy to solve using constraint programming. The program takes less than 20 lines of codes. We used the very useful matrix_util library for the representation of the board, which allows to directly feed the columns and rows to the ic_global sumlist constraint. Again, the well-known alldifferent constraint is used.

```
1 :- lib(ic).
2 :- lib(ic_global).
3 :- lib(matrix_util).
4 :- import sumlist/2, alldifferent/1 from ic_global.
5
6 %a survo puzzle!
7 survo([X,Y], Board) :-
8   length(Y,N),length(X,M), Max is N*M,
9   matrix(N, M, Board, Cols),
```

```
10    flatten(Board,B),
11    B :: 1..Max,
12    alldifferent(B),
13    (foreach(Row,Board), foreach(RowSum,Y) do
14      sumlist(Row,RowSum)
15    ),
16    (foreach(Col,Cols), foreach(ColSum,X) do
17      sumlist(Col,ColSum)
18    ),
19    term_variables(B,Vars),
20    labeling(Vars).
```

8 Experiments

We also tested our program on a few examples. In particular, we tested it especially on open Survo puzzles (where no hints are given [12]). We refer to the full program given in Appendix A for the test data. It is clear that this is a very efficient approach and even puzzles which are almost completely impossible for humans, are solved easily (puzzle 7 did take 40 s but the rating on the `http://www.survo.fi/puzzles/` did confirm that this was an extremely difficult one).

```
*** Puzzle 1, Horizontal [21, 10, 18, 29],  Vertical [24, 15, 39]
Took 0.03 seconds to solve this one:
[7, 3, 5, 9]
[4, 1, 2, 8]
[10, 6, 11, 12]

*** Puzzle 2, Horizontal [21, 10, 18, 29],  Vertical [24, 15, 39]
Took 0.00 seconds to solve this one:
[7, 3, 5, 9]
[4, 1, 2, 8]
[10, 6, 11, 12]

*** Puzzle 3, Horizontal [51, 42, 26, 17],  Vertical [51, 36, 32, 17]
Took 0.07 seconds to solve this one:
[15, 16, 12, 8]
[14, 11, 7, 4]
[13, 10, 6, 3]
[9, 5, 1, 2]

*** Puzzle 4, Horizontal [42, 51, 63, 21, 33],  Vertical [38, 63, 83, 26]
Took 4.73 seconds to solve this one:
[8, 9, 14, 2, 5]
[13, 16, 18, 6, 10]
```

```
[17, 19, 20, 12, 15]
[4, 7, 11, 1, 3]

*** Puzzle 5, Horizontal [49, 31, 38, 18],  Vertical [17, 54, 40, 25]
Took 0.10 seconds to solve this one:
[8, 3, 4, 2]
[16, 13, 15, 10]
[14, 9, 12, 5]
[11, 6, 7, 1]

*** Puzzle 6, Horizontal [48, 54, 30, 35, 43],  Vertical [17, 38, 85, 70]
Took 10.46 seconds to solve this one:
[4, 7, 1, 2, 3]
[9, 10, 5, 6, 8]
[19, 20, 13, 15, 18]
[16, 17, 11, 12, 14]

*** Puzzle 7, Horizontal [48, 30, 69, 40, 23],  Vertical [62, 27, 80, 41]
Took 39.30 seconds to solve this one:
[14, 9, 19, 12, 8]
[6, 2, 13, 5, 1]
[18, 15, 20, 16, 11]
[10, 4, 17, 7, 3]

*** Puzzle 8, Horizontal [41, 19, 51, 33, 66],  Vertical [72, 63, 19, 56]
Took 4.38 seconds to solve this one:
[15, 7, 17, 13, 20]
[12, 6, 16, 10, 19]
[3, 1, 4, 2, 9]
[11, 5, 14, 8, 18]

*** Puzzle 9, Horizontal [24, 33, 38, 57, 58],  Vertical [20, 44, 65, 81]
Took 6.98 seconds to solve this one:
[1, 2, 3, 5, 9]
[4, 6, 7, 14, 13]
[8, 10, 12, 18, 17]
[11, 15, 16, 20, 19]

*** Puzzle 10, Horizontal [12, 17, 28, 37, 40, 45, 52],  Vertical [50, 67, 114]
Took 0.87 seconds to solve this one:
[1, 3, 5, 7, 8, 11, 15]
[2, 4, 6, 12, 13, 14, 16]
[9, 10, 17, 18, 19, 20, 21]
```

References

[1] A. Anagnostopoulos, L. D. Michel, P. Van Hentenryck, and Y. Vergados. A simulated annealing approach to the traveling tournament problem. *Journal of Scheduling*, 9:177–193, 2006.

[2] T. Benoist, F. Laburthe, and B. Rottembourg. Lagrange relaxation and constraint programming collaborative schemes for travelling tournament problems. In *In CP-AI-OR'2001, Wye College*, pages 15–26, 2001.

[3] D. G. Bobrow and B. Raphael. New programming languages for artificial intelligence research. *ACM Comput. Surv.*, 6:153–174, September 1974.

[4] L. De Raedt, T. Guns, and S. Nijssen. Constraint programming for data mining and machine learning. In *AAAI*, pages 1671–1675, 2010.

[5] K. Easton, G. Nemhauser, and M. Trick. The traveling tournament problem description and benchmarks. In *Proceedings of the 7th International Conference on Principles and Practice of Constraint Programming*, volume 2239, pages 580–584, 2001.

[6] K. Easton, G. L. Nemhauser, and M. A. Trick. Solving the travelling tournament problem: A combined integer programming and constraint programming approach. In *PATAT*, pages 100–112, 2002.

[7] M. Fürer and S. Prasad Kasiviswanathan. Approximately counting embeddings into random graphs. In *Proceedings of the 11th international workshop, APPROX 2008, and 12th international workshop, RANDOM 2008 on Approximation, Randomization and Combinatorial Optimization: Algorithms and Techniques*, APPROX '08 / RANDOM '08, pages 416–429, Berlin, Heidelberg, 2008. Springer-Verlag.

[8] A. Kimmig, B. Demoen, L. De Raedt, V. Costa, and R. Rocha. On the implementation of the probabilistic logic programming language Problog. *Theory Pract. Log. Program.*, 11:235–262.

[9] S. Kirkpatrick, J. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[10] H. J. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. B. Scherl. Golog: A logic programming language for dynamic domains. *Journal of Logic Programming*, 31, 1997.

[11] S. Mustonen. On certain cross sum puzzles, 2006.

[12] S. Mustonen. Enumeration of uniquely solvable open Survo puzzles, 2007.

[13] G. Neumann. Programming Languages in Artificial Intelligence. In Bentley and Bidgoli, editors, *Encyclopedia of Information Systems*. Academic Press, San Diego, 2002.

[14] L. Quan. Investigate parameters of Fürer's algorithm. Capita Selecta AI: Task 3, 2011.

[15] L. Quan and P. Tanghe. Regel-gebaseerde regressie. Capita Selecta AI: Task 2, 2011.

[16] L. Quan and P. Tanghe. Simulated annealing voor TTP (TTSA). Capita Selecta AI: Task 1, 2011.

[17] A. Shapiro, D. Dentcheva, and R. Andrzej. *Lectures on stochastic programming: modeling and theory*, volume 9 of *MPS/SIAM Series on Optimization*. SIAM, 2009.

[18] M. Trick. Challenge traveling tournament problems, 2011. `http://mat.gsia.cmu.edu/TOURN/`.

[19] M. A. Trick. Integer and constraint programming approaches for round-robin tournament scheduling. In *PATAT*, pages 63–77, 2002.

[20] G. Van den Broeck, I. Thon, M. V. Otterlo, and L. D. Raedt. DTProbLog: A Decision-Theoretic Probabilistic Prolog. In M. Fox and D. Poole, editors, *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11–15, 2010*. AAAI Press, 2010.

[21] P. Van Hentenryck and Y. Vergados. Traveling tournament scheduling: A systematic evaluation of simulated annealing. In *CPAIOR*, pages 228–243, 2006.

[22] J. Vennekens, S. Verbaeten, and M. Bruynooghe. Logic programs with annotated disjunctions. In *In Proc. Int'l Conf. on Logic Programming*, pages 431–445. Springer, 2004.

[23] Wikipedia. Survo puzzle — Wikipedia, the free encyclopedia, 2011. `http://en.wikipedia.org/wiki/Survo_Puzzle` [last accessed: May 31, 2011].

A Survo puzzle, ECLiPSe

```
:- lib(ic).
:- lib(ic_global).
:- lib(matrix_util).
:- import sumlist/2, alldifferent/1 from ic_global.


%a survo puzzle! see http://en.wikipedia.org/wiki/Survo_Puzzle
survo([X,Y], Board) :-
  length(Y,N),length(X,M), Max is N*M,
  matrix(N, M, Board, Cols),     %check the format of Board or create it and also make list of cols
  flatten(Board,B),
  B :: 1..Max,
  alldifferent(B),

  (foreach(Row,Board), foreach(RowSum,Y) do
    sumlist(Row,RowSum)
  ),
  (foreach(Col,Cols), foreach(ColSum,X) do
    sumlist(Col,ColSum)
  ),

  term_variables(B,Vars),
  labeling(Vars).


%% solves all example puzzles, should take ~ 1 min
test :- findall((Id,P,[X,Y]),problem(Id,P,[X,Y]), List),
    ( foreach((Id,P,[X,Y]),List) do
      printf("*** Puzzle %d, Horizontal %w,  Vertical %w\n",[Id,X,Y]),
      statistics(times, [T0|_]),  survo([X,Y],P), statistics(times, [T1|_]),
        Secs is T1-T0,
        printf("Took %.2f seconds to solve this one:\n", [Secs]),
      pretty_print(P)
    ).

%%%some test data
%problem(Id, Initial, [Horizontal, Vertical])

problem(1, _, [[21,10,18,29], [24,15,39]]).
problem(2, P, [[21,10,18,29], [24,15,39]]) :-
  P = [ [7,_,5,_],
      [_,1,_,8],
      [_,_,11,_]
    ].

problem(3, _, [[51,42,26,17], [51,36,32,17]]).

```

```
48 %some more difficult http://www.survo.fi/puzzles/
49 problem(4, _, [[42,51,63,21,33],[38,63,83,26]]).
50 problem(5, _, [[49,31,38,18],[17,54,40,25]]).
51 problem(6, _, [[48,54,30,35,43],[17,38,85,70]]).
52 problem(7, _, [[48,30,69,40,23],[62,27,80,41]]). %this one is
       very difficult
53 problem(8, _, [[41,19,51,33,66],[72,63,19,56]]).
54 problem(9, _, [[24,33,38,57,58],[20,44,65,81]]).
55 problem(10, _, [[12,17,28,37,40,45,52],[50,67,114]]).
56
57 %%
58
59 pretty_print(P) :-
60   ( foreach(Row,P) do
61     writeln(Row)
62   ), nl.
```