CSCE 421/821: Foundations of Constraint Processing, Fall2004

List of Projects

Berthe Y. Choueiry
Constraint Systems Laboratory
Department of Computer Science and Engineering
University of Nebraska-Lincoln
`choueiry@cse.unl.edu`
September 15, 2004

Distributed: Wednesday, September 15, 2004

Project selection: Monday, October 4th, 2004 (by `handin`)

Progress reports: Friday, November 5th, 2004 (by `handin`)

Final reports: Friday, December 3rd 2004 (paper and by `handin`)

Presentations: M/W/F, December 6–10, 2004 (evening sessions scheduled as necessary)

Code and slides (when applicable): Friday, December 10, 2004 (by `handin`)

Notes: *You must clearly acknowledge help received from anyone*. Always acknowledge sources of information (URL, book, class notes, etc.).

Contents

---

1 Guidelines

This is a non-exhaustive list of possible topics for semester projects. If you have an idea for a project, do not hesitate to discuss it with the instructor. There are three main categories to choose from:

1. The search challenge described in Section 2. This requires familiarity with Common Lisp.
2. A project from one of the following alternatives. These include the following main alternatives:
   - Implement and evaluate an algorithm, Sections 3.
   - Study modeling and problem formulation, Section 4.
   - Model and solve a (simple) practical problem, Section 5.
   - Investigate an advanced theoretical concept, Section 6.
3. Conduct a critical literature survey, Section 7.

General guidelines:

- Again, you are encouraged to design your own project proposal and discuss it with instructor.
- Some projects are substantial enough to be conducted in teams of two or three students. When this is the case, *each student will have* to provide the instructor with an evaluation of the performance of his/her team partners.
- The same project may be chosen by more than one person or team. So, do not rush to 'reserve' yourself a project. If a project is selected by more than one person or team, we will carry out a comparison of the approach and results.
- Collaborate with a research assistant. Contact a research assistant and convince him/her to invest in you and work with you. Define with him/her a project to work on. He/she will provide mentoring and supervision in tight collaboration with the instructor.

---

2 Search competition

In the Constraint Systems Laboratory, we have modeled as a CSP the problem of assigning Graduate Teaching Assistants (GTAs) to courses in the department [Glaubius 2001; Glaubius and Choueiry 2002; Zou and Choueiry 2003; Lim *et al.* 2004; Guddeti and Choueiry 2004]. We have encoded the model (i.e., data structures and constraints) in Common Lisp. Student are requested to design and implement their own problem-solving strategy to handle this over-constrained optimization problem. Results will be compared according to a predefined set of optimization criteria. We will provide a specification of the problem, a collection of data, and a (sketchy) manual of the main functions for checking consistency of constraints. Students can investigate any of the following approaches:

1. Learning techniques for the development of algorithm selection: we have (at least) 5 search techniques implemented. The goal is to build a mechanis of portofolio selection based on the performance of these algorithms. Building on the work of [Leyton-Brown *et al.* 2003b; 2003a; for Automatic Algorithm Portfolio Seclection 2004].
2. Constructive search.

3. Iterative repair (e.g., hill-climbing and tabu search).

4. Auction and market-based algorithms.

5. A constructive or local search technique that exploits symmetries (i.e., interchangeability).

This project requires the ability to deal with (i.e., program in or interface with) Common Lisp. It also requires a sense of creativity for designing a new problem-solving strategy, or choosing and adapting one from the literature. Since the current prototype is the product of research activities, the robustness and stability of the code are not guaranteed.

3 Experimental evaluation of advanced algorithms

Every implementation should be tested on a real-world problem (when available), randomly generated problems, or both. Results should be reported in terms of nodes visited, constraint checks, CPU time, and other applicable criteria. Generators for random binary and non-binary CSPs are available. Tests should be conducted for various values of constraint tightness and density, and results should be averaged for at least 50 problem instances per measurement point. Details should be discussed with instructor on a case by case basis.

1. Study, implement, and evaluate the algorithm for finding super solutions of Hebrard asynchronous [Hebrard *et al.* 2004].

2. Study, implement, and evaluate the algorithm for asynchronous backtracking by Maestre and Bessière [Maestre and Bessi‘ere 2004].

3. Study the effect of the choice of the value of the noise parameter in random walk in local search on the performance of search [Wallace 1996].

4. Implement and compare the performance of the following algorithms for path consistency: `PC-2` [Mackworth 1977], `DPC` [Dechter 2003], `PPC` [Bliek and Sam-Haroud 1999], and $\triangle$-`PPC` [Xu 2003].

5. Design, implement, and test a local search algorithm for solving the TCSP. Test on randomly generated instances from the Constraint Systems Lab. Verify the existence of a phase transition and compare to the results obtained by Xu [Xu 2003].

6. Investigate the existence and occurrence of the phase transition phenomenon when solving a CSP with intelligent backtracking algorithms (i.e., backjumping and conflict-directed backtracking). Compare the performance of these algorithms with that of backtrack-search with forward-checking (FC).

7. Implement the algorithm of Inferred Disjunctive Constraint of [Freuder 1993] and test it on randomly generated problems. Compare its performance with that of backtrack-search with forward-checking (FC).

8. Implement the decomposition algorithm based on tree clustering described in [Dechter and Pearl 1989]. Run experiments on randomly generated binary CSPs and report performance of the resulting mechanism.

9. Implement the arc-consistency algorithm for `all-diff` constraints and use it in a backtrack search procedure with forward-checking to solve resource allocation problems[1]. Report performance of the resulting mechanism for finding one solution and compare it with that of AC-3.

10. Implement the constraint propagation algorithm for interval-based temporal knowledge [Allen 1981]. Find a practical application suitable for such a model then design and conduct experiments.

11. Implement, test, and evaluate the asynchronous search for distributed CSPs of [Yokoo 1995].

12. Implement, test, and evaluate the asynchronous search for distributed CSPs of [Silaghi *et al.* 2000].

13. Study the new algorithms for incremental triangulations of graphs of [Noubir 2003] and [Berry *et al.* 2003] and compare their performance with that of the algorithm that computes triangulation from scratch [Kjaerulff 1990].

14. Combine dynamic bundling techniques [Choueiry and Davis 2002; Lal and Choueiry 2003] with an algorithm for solving the Temporal Constraint Satisfaction Problem (TCSP), such as $\triangle$STP-TCSP of [Xu and Choueiry 2003b].

15. Using the idea $\triangle$AC of [Xu and Choueiry 2003a], implement and evaluate forward-checking for solving the Temporal Constraint Satisfaction Problem (TCSP) using $\triangle$STP-TCSP of [Xu and Choueiry 2003b].

16. Study, implement, and evaluate dynamic variable-ordering heuristics for solving the temporal Constraint Satisfaction Problem (TCSP) using $\triangle$STP-TCSP of [Xu and Choueiry 2003b].

17. Study, implement, and evaluate the backtrack search for solving the Disjunctive Temporal Problem (DTP) proposed in [Tsamardinos and Pollack 2003].

18. Study, implement, and evaluate the algorithm for decomposing CSPs of [Chmeiss *et al.* 2003].

4 Modeling

1. Consider a number of common puzzles (e.g., Zebra, Aliens Tiles Puzzles, Quasigroups, Latin Squares, and Golomb ruler). Model them as non-binary CSPs. Implement a search mechanism with a non-binary FC (nFC) mechanisms [Bessière *et al.* 1999] to solve them. Report and compare results.

2. Consider a number of benchmark problems available from the CSP Library `www.csplib.org`. Model them as non-binary CSPs. Implement a search mechanism with a non-binary FC (nFC) mechanisms [Bessière *et al.* 1999] to solve them. Report and compare results.

5 Applications

1. *Telecom.* Study, implement, and test the algorithm for resource allocation in telecommunication networks of [Frei and Faltings 1999].

2. *Manufacturing.* Study, implement, and test strategies for job-shop scheduling of [Caseau and Laburthe 1994; 1996].

[1]Data will be provided by the instructor.

3. *Manufacturing.* Study, implement, and test strategies for job-shop scheduling as a CSP of [Cheng and Smith 1997].

4. *Manufacturing.* Study, implement, and test strategies for scheduling with inventory of [Beck 2000] or inventory management of [Rodosek *et al.* 1997].

5. *Bioinformatics.* Choose one paper from the Special Issue on Bioinformatics of the Constraints Journal that models and solves a problem in bioinformatics as a constraint satisfaction problem. Implement the technique described and report your experience.

6 Research

1. *Relationship between CSPs and Database Theory.* Study the work of Feder and Vardi.

2. *Use of Database techniques in CSPs.* Study the work of Gottlob *et al.*

3. *Use of Database techniques in CSPs.* Study how the join ordering strategies using dynamic programming can be exploited for finding all the solutions to a non-binary constraint satisfaction problem. Implement, test, and evaluate the algorithm.

4. *Generation of random solutions.* Study, implement and test the method for generating solutions to a CSP uniformly at random of [Dechter *et al.* 2002].

5. *Bundling strategies.* Modify the MAC algorithm to exploit interchangeability. Implement your modification, test it on binary CSPs and compare results with the non-modified algorithm. (Initial code exists in Common Lisp.)

6. *Disjunctive decomposition.* Study the following disjunctive decomposition strategies: Out Failure [Freuder and Hubbe 1995], Inferred Disjunctive Constraint [Freuder 1993], and co-microstructure based decomposition [Choueiry and Noubir 1997]. Implement a combination of these on binary CSPs and analyze results.

7 Literature review

Conduct a critical review and a synthesis of an area where constraints are studied or applied, such as:

1. The various encodings of SAT as a CSP (there are up to 7 of them!).
   This topic could also be made into a project for evaluating and comparing the performance of search on the CSPs resulting from the various encodings of a SAT instance (over a library of SAT instances).

2. Symmetric CSPs.

3. Backbone in SAT and CSPs.

4. Use of SAT and constraints in Model Checking.

5. Soft constraints and preferences in CSPs: CP-nets.

6. Temporal reasoning.

7. Distributed CSPs.

8. Numeric (a.k.a. continuous) CSPs.

9. Relations between CSPs and belief networks.

8 List of papers for summaries or presentations

Here is a list of papers you may want to study for a summary:

- Semiring and soft constraints for diagnosis [Sachenbacher and Williams 2004].
- The paper in conditional interchangeability and substitutability of [Zhang and Freuder 2004].
- The paper on variable ordering heuristics by Beck et al. [Beck *et al.* 2003]. (You may find it useful to read the short ones too: [Beck *et al.* 2004a; 2004b].

References

Allen, James F. 1981. An interval based representation of temporal knowledge. In *Proc. of the 7 th IJCAI*, Vancouver, Canada. 221–226.

Beck, J. Christopher; Prosser, Patrick; and Wallace, Richard J. 2003. Toward Understanding Variable Ordering Heuristics for Constraint Satisfaction Problems. In *Proceedings of the Fourteenth Irish Artificial Intelligence and Cognitive Science Conference (AICS03)*.

Beck, J. Christopher; Prosser, Patrick; and Wallace, Richard J. 2004a. Failing First: An Update. In *Proceedings of the Sixteeth European Conference on Artificial Intelligence*. 959–960.

Beck, J. Christopher; Prosser, Patrick; and Wallace, Richard J. 2004b. Variable Ordering Heuristics Show Promise. In *the Tenth International Conference on Principles and Practice of Constraint Programming (CP'04)*.

Beck, J. Christopher 2000. Heuristics for Constraint-Directed Scheduling with Inventory. In *Proceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling (AIPS-2000)*, Breckenridge, CO. 22–31.

Berry, Anne; Heggernes, Pina; and Villanger, Yngve 2003. An On-Line Incremental Approach for Dynamically Maintaining Chordal Graphs. Available from instructor.

Bessière, Christian; Meseguer, Pedro; Freuder, Eugene C.; and Larrosa, Javier 1999. On forward checking for non-binary constraint satisfaction. In *Principles and Practice of Constraint Programming (CP'99)*. 88–102.

Bliek, Christian and Sam-Haroud, Djamilla 1999. Path Consistency for Triangulated Constraint Graphs. In *Proc. of the 16 th IJCAI*, Stockholm, Sweden. 456–461.

Caseau, Yves and Laburthe, Francois 1994. Improved CLP Scheduling with Task Intervals. In Hentenryck, Pascalvan, editor 1994, *Proceedings of the 11th International Conference on Logic Programming, ICLP'94*. The MIT press.

Caseau, Yves and Laburthe, Francois 1996. Cumulative scheduling with task intervals. In *Joint International Conference and Symposium on Logic Programming*. 363–377.

Cheng, Cheng-Chung and Smith, Stephen F. 1997. Applying Constraint Satisfaction Techniques to Job Shop Scheduling. *Annals of Operations Research* 70:327—357.

Chmeiss, Assef; Jégou, Philippe; and Keddar, Lamia 2003. On a Generalization of Triangulated Graphs for Domains Decomposition of CSPs. In *Proc. of the 18 th IJCAI*, Acapulco, Mexico. 203–208.

Choueiry, Berthe Y. and Davis, Amy M. 2002. Dynamic Bundling: Less Effort for More Solutions. In Koenig, Sven and Holte, Robert, editors 2002, *5th International Symposium on Abstraction, Reformulation and Approximation (SARA 2002)*, volume 2371 of *Lecture Notes in Artificial Intelligence*. Springer Verlag. 64–82.

Choueiry, Berthe Y. and Noubir, Guevara 1997. Using Complete No-Good Sets to Build a Disjunctive Decomposition Scheme for Discrete Constraint Satisfaction Problemsxo. In *10th International FLAIRS conference (FLAIRS'97)*, Daytona Beach, FL. 40–46. Revised version available as technical report KSL-98-23, `ksl-web.stanford.edu/KSL_Abstracts/KSL-98-23.html`.

Dechter, Rina and Pearl, Judea 1989. Tree Clustering for Constraint Networks. *Artificial Intelligence* 38:353–366.

Dechter, Rina; Kask, Kalev; Bin, Eyal; and Emek, Roy 2002. Generating Random Solutions for Constraint Satisfaction Problems. In *Proc. of AAAI-2002*, Edmoton, Alberta, Canada. 15–21.

Dechter, Rina 2003. Constraint Processing. Manuscript, forthcoming.

Automatic Algorithm Portfolio Seclection, Learning Techniquesfor 2004. Failing First: An Update. In *Proceedings of the Sixteeth European Conference on Artificial Intelligence*. 475–479.

Frei, Christian and Faltings, Boi 1999. Resource allocation in networks using abstraction and constraint satisfaction techniques. In Jaffar, Joxan, editor 1999, *Fifth International Conference on Principles and Practice of Constraint Programming (CP'99)*, Alexandria, Virginia, USA. Lecture Notes in Computer Science, Vol. 1713, Springer-Verlag. 204–218.

Freuder, Eugene C. and Hubbe, Paul D. 1995. Extracting Constraint Satisfaction Subproblems. In *Proc. of the 14 th IJCAI*, Montréal, Québec, Canada. 548–555.

Freuder, Eugene C. 1993. Using Inferred Disjunctive Constraints to Decompose Constraint Satisfaction Problems. In *Proc. of the 13 th IJCAI*, Chambéry, France. 254–260.

Glaubius, Robert and Choueiry, Berthe Y. 2002. Constraint Modeling and Reformulation in the Context of Academic Task Assignment. In *Working Notes of the Workshop on Modelling and Solving Problems with Constraints, ECAI 2002*, Lyon, France.

Glaubius, Robert 2001. A Constraint Processing Approach to Assigning Graduate Teaching Assistants to Courses. Undergraduate Honors Thesis. Department of Computer Science and Engineering, University of Nebraska-Lincoln.

Guddeti, Venkata Praveen and Choueiry, Berthe Y. 2004. An Empirical Study of a New Restart Strategy for Randomized Backtrack Search. In *Workshop on Immediate Applications of CSPs (CP 04)*, Toronton, Canada.

Hebrard, Emmanuel; Hnich, Brahim; and Walsh, Toby 2004. Super solutions in constraint programming. In *Proceedings CPAIOR'04*, Nice, France. 4c.ucc.ie/˜ehebrard/stuff/cpaior2004.pdf.

Kjaerulff, U. 1990. Triagulation of Graphs - Algorithms Giving Small Total State Space.

Lal, Anagh and Choueiry, Berthe Y. 2003. Dynamic Detection and Exploitation of Value Symmetries for Non-Binary Finite CSPs. In *Third International Workshop on Symmetry in Constraint Satisfaction Problems (SymCon'03)*, Kinsale, County Cork, Ireland. 112–126.

Leyton-Brown, K.; Nudelman, E.; Andrew, G.; McFadden, J.; and Shoham, Y. 2003a. Boosting as a metaphor for algorithm design (working paper). http://www.cs.ubc.ca/˜kevinlb/papers/boosting.ps.

Leyton-Brown, K.; Nudelman, E.; Andrew, G.; McFadden, J.; and Shoham, Y. 2003b. Boosting as ametaphor for algorithm design. In *Constraint Programming*.

Lim, Ryan; Guddeti, Venkata Praveen; and Choueiry, Berthe Y. 2004. An Interactive System for Hiring and Managing Graduate Teaching Assistants. In *Conference on Prestigious Applications of Intelligent Systems (ECAI 04)*, Valencia, Spain. 730–734.

Mackworth, Alan K. 1977. Consistency in Networks of Relations. *Artificial Intelligence* 8:99–118.

Maestre, Arnold and Bessi‘ere, Christian 2004. Improving Asynchronous Backtracking for Dealing with Complex Local Problems. In *European Conference on Artificial Intellifence (ECAI 04)*, Valencia, Spain. 206–210.

Noubir, Guevara 2003. Personal communication.

Rodosek, R.; Kokeny, T.; Caseau, Y.; and LePape, C. 1997. Creating hybrid solutions for inventory management problems.

Sachenbacher, Martin and Williams, Brian C. 2004. Diagnosis as Semiring-Based Constraint Optimization. In *European Conference on Artificial Intellifence (ECAI 04)*, Valencia, Spain. 873–877.

Silaghi, Marius-Calin; Sam-Haroud, D.; and Faltings, B. 2000. Asynchronous search with aggregations. In *AAAI2000*, Austin.

Tsamardinos, Ioannis and Pollack, Martha E. 2003. Efficient Solution Techniques for Disjunctive Temporal Reasoning Problems. *Artificial Intelligence*. In press.

Wallace, Richard J. 1996. Analysis of heuristic methods for partial constraint satisfaction problems. In *Principles and Practice of Constraint Programming*. 482–496.

Xu, Lin and Choueiry, Berthe Y. 2003a. An Approximation of Generalized Arc-Consistency for TCSPs. In *Working notes of the Workshop on Spatial and Temporal Reasoning (IJCAI 03)*, Acapulco, Mexico. 27–32.

Xu, Lin and Choueiry, Berthe Y. 2003b. Improving Backtrack Search for Solving the TCSP. In *Principles and Practice of Constraint Programming (CP 03)*, Kinsale, County Cork, Ireland. LNCS 2833, Springer Verlag. 754–768.

Xu, Lin 2003. Consistency Methods for Temporal Reasoning. Master’s thesis, Department of Computer Science and Engineering, University of Nebraska-Lincoln, Lincoln, NE. May 2003.

Yokoo, Makoto 1995. Asynchronous weak-committment search for solving distributed constraint satisfaction problems. In *Principles and Practice of Constraint Programming (CP’95)*, volume 976 of *Lecture Notes in Computer Science*. Springer Verlag. 88–102.

Zhang, Yuanlin and Freuder, Eugene C. 2004. Conditional interchangeability and substitutability. In *Fourth International Workshop on Symmetry in Constraint Satisfaction Problems (SymCon’04)*, Toronto, Canada.

Zou, Hui and Choueiry, Berthe Y. 2003. Characterizing the Behavior of a Multi-Agent Search by Using it to Solve a Tight, Real-World Resource Allocation Problem. In *Workshop on Applications of Constraint Programming*, Kinsale, County Cork, Ireland. 81–101.