# An Interactive System for Hiring and Managing Graduate Teaching Assistants

**Ryan Lim, Venkata Praveen Guddeti, and Berthe Y. Choueiry** [1]

**Abstract.** In this paper, we describe a system for managing the hiring and assigning of Graduate Teaching Assistants (GTAs) to academic tasks based on the GTAs qualifications, preferences, and availability. This system is built using Constraint Processing techniques and is operated through web-based interfaces. Various versions of the prototype system have been in actual use since Fall 2001 and have yielded a significant improvement in the quality and stability of the final assignments in our department and a reduction of the workload and frustration of the administrators involved in this task. This paper describes the motivation and practical significance of this system, the design and functionalities of its components, and the teaching and research opportunities it has enabled.

## 1 INTRODUCTION

This paper describes a system we designed and implemented for the management of Graduate Teaching Assistants (GTAs) [3] in the Department of Computer Science & Engineering (CSE) of the University of Nebraska-Lincoln. The task is to assign GTAs, based on their qualifications and availability, to academic tasks such as grading, supervising labs and recitations, and teaching introductory classes. This application is a critical and arduous responsibility that the department's administration has to drudge through every semester. Typically, every semester, the department has about 70 different academic tasks and can hire between 25 and 40 GTAs. The problem is often tight and sometimes over-constrained. In the past, this task has been performed by hand by members of the staff and faculty. Tentative schedules were iteratively refined and updated based on feedback from other faculty members and the GTAs themselves, in a tedious and error-prone process lingering over 3 weeks. It was quite common for the final hand-made assignments to contain a number of conflicts and inconsistencies, which negatively affects the quality of our program. For instance, when a course is assigned a GTA with little knowledge of the subject matter, the course's instructor has to take over a large portion of the GTA's job and the GTA has to invest considerable effort to adjust to the situation. Moreover, students in the course may receive diminished feedback and unfair grading.

We have built a web-based interface that streamlines GTA applications and allows candidates to specify their preferences for tasks on a scale ranging from 0 (cannot handle) to 5 (best choice). Further, we have implemented a number of functionalities, based on constraint processing techniques, that assist the human manager in generating solutions automatically or interactively. The modeling efforts started in Spring 2001. Various versions of the prototype have been used every semester since Fall 2001. This system has effectively reduced the number of obvious conflicts, with positive effects on the quality of our academic program. It has also decreased the amount of time and effort spent on making the assignment and gained the approval and satisfaction of our staff, faculty and student body.

Section 2 describes the design of the system. Section 3 summarizes the benefits of our endeavor, both for administration of the department and also for training students in AI modeling and research. Finally, Section 4 and Section 5 draws directions for future research and concludes this paper.

[1] Constraint Systems Laboratory, Computer Science and Engineering, University of Nebraska-Lincoln, Lincoln NE 68588-0115 email: {rlim, vguddeti, choueiry}@cse.unl.edu

## 2 SYSTEM ARCHITECTURE

The current system has the components shown in Fig. 1: web-interfaces for data acquisition from a human manager and individual applicants, a relational database for storing the collected data, a number of search algorithms, and the ability to generate reports. Current developments include secure interfacing with university, college and department databases, visualization widgets for supporting the manager in decision making, and an integrated system for algorithm management and collaborative strategies. The implementation is heterogeneous: the database is a MySQL database, the web-interfaces are implemented in PHP, and the search algorithms are in Common Lisp and C++. Below we describe the implemented components. Sections 2.1 to 2.3 focus on the input and manipulation of data and have, for now, little to do with constraint processing. However, they are essential for implementing and running the decision-making algorithms and for designing intelligent techniques for data visualization. Section 2.4 discusses the constraint model and the interactive and automated search algorithms.

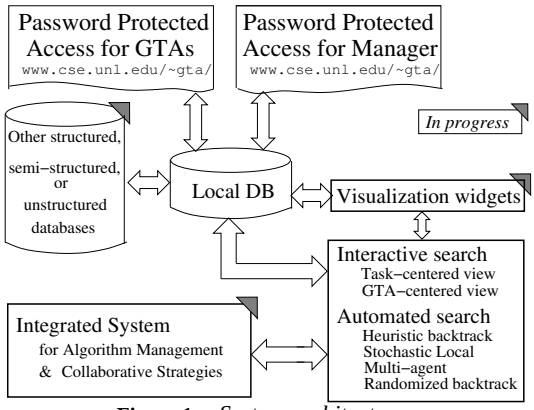


**Figure 1.** *System architecture.*

## 2.1 MySQL Database Server

We designed a MySQL database to store the information entered by the applicants and the manager. The schedule of classes is semi-automatically retrieved from a departmental database. All the components of the system use the database directly or indirectly (Fig. 1) to store and retrieve data. We foresee a secure university-wide integration with (more or less) structured databases, which constitutes an invaluable opportunity for extending the use of constraint-processing techniques to database access security.

## 2.2 Web-interface for GTAs

Graduate students can at any time apply for consideration and update their record. A deadline for receiving applications can optionally be enforced. Accounts are password protected. It is crucial that the system not be restricted to applicants who already have accounts in the department. Indeed, admitted graduate students from foreign countries need to be able to register on the system and input their academic data and preferences over the Internet prior to arrival. The interface offers four main modes.

The first one is a form for inputing the academic record of applicants, which was formerly collected by the department on paper. This includes the number of semesters supported so far, the level of support, the current advisor, previous teaching experience, GPA, English proficiency level, ITA qualification[2], the list of course deficiencies, etc. The second option displays the list of courses offered during the semester and their meeting times. For each course, an applicant can specify his/her preferences: '5 Best choice,' '4 Favorite,' '3 Qualified,' '2 Able to handle,' '1 Avoid if possible,' '0 Cannot handle.' The student can check a box indicating that he/she is enrolled in given course, which automatically sets up the preference value to 0. In all other cases where the applicant specifies a preference value of 0, the system prompts the applicant to provide a justification for the inability to handle the course. A preference value is set to 3 by default to reduce the applicants' burden in specifying preferences. At the bottom of the window, a permanent legend reminds the students of the meaning of the different preference values. The third functionality provides students with direct access to the official course descriptions. This is useful for incoming students who have not yet joined the department. Finally, the last functionality is an optional survey, requesting applicants' comments and their feedback on the ease of navigation and data entry.

The first two modes (i.e., academic record and teaching preferences) display the time-stamp of the last modifications made by the applicant. Importantly, every time an applicant modifies his/her record or preference selections, a message confirming the new data input is sent via email to the applicant for his/her personal record.

## 2.3 Web-interface for manager

The interface for the manager permanently displays, in a horizontal frame on the top of the page, five buttons representing the operational modes for (1) managing the GTAs, (2) managing the classes, (3) performing interactive selections, (4) running search algorithms, and (5) making server-level commands. These five modes are controlled by a semester selector, set up by default to the most recent semester in the database. When any of these 5 modes is selected, additional buttons for selecting functionalities specific to the mode appear in a vertical frame in the left margin. These buttons are displayed as long as the mode is active.

The first mode allows the manager to manage the GTAs in the system, i.e., those that have applied for the specified semester and those that are hired for that semester. The list of GTAs considered at any point in time can also be filtered by any number of predefined selection criteria based on the applicants' performance history via a graphical template for selective queries. The manager has the ability to hire, release, and specify the hiring percentage (i.e., GTA capacity) of a particular applicant, and to view and generate PDF reports about any number of the listed applicants. Further, the manager can send an email message to any subset of these applicants. Finally, the manager can delete the record of any applicant or add new ones.

The second mode allows the manager to manage the classes in a semester, specifying course loads, their types (e.g., grading, lab, or lecture), and various dependencies among courses. Fig. 2 shows a screen shot of a page of the second mode. In this mode, the manager can also enforce the assignments of GTAs to classes to satisfy some requirement external to the system. For each class, a pull-down menu lists all hired GTAs (regardless of whether they are available, appropriate for the task, etc.) from which manager can select. Such pre-assignments are considered hard, can be enforced only through this specific page, and cannot be undone by the interactive (Section 2.4.2) or automated (Section 2.4.3) search facilities.

The third mode is currently thought to be the most useful by our users because it assists the manager in making interactively individual or group assignments manually in an efficient way. The manager remains in total control of the decisions, but is received of the burden of keeping track of the consistency among decision. The system offers a dual perspective: a task-centered view and a GTA-centered view. We use constraint-based techniques to display the GTAs (alternatively, courses) available and not for each course (alternatively, GTA), and to propagate the effects of the manager decisions on the remaining open decisions, see Section 2.4.2.

The fourth mode offers several search algorithms for automatically solving the problem. These algorithms consider the pre-assignments made in the second mode as hard constraints, but do not (yet) take into account the interactive decisions made in the third mode. These algorithms run independently whenever selected by the user.

Finally, the last mode provides the manager with system-level functionalities such as access to the spawned processes.

## 2.4 Search algorithms

In this section, we describe the techniques of Constraint Processing that we used or developed for this application. The details of these techniques can be found in the listed references. The main contributions encompass a constraint model (Section 2.4.1), the application of consistency techniques for interactive decision making (Section 2.4.2), implementation of various heuristic and stochastic search strategies (Section 2.4.3), and characterization of the relative performance of these strategies under our particular setting (Section 2.5).

### *2.4.1 Constraint-based model*

The task is to assign GTAs according to various criteria (considered as hard constraints) to academic tasks. The problem is almost always tight and often over-constrained, but this is not known a priori. The goal is to cover as many classes as possible while maximizing the

[2] International Teaching Assistant (ITA) is a university monitored training that foreign students are required to complete to qualify for classroom instruction.

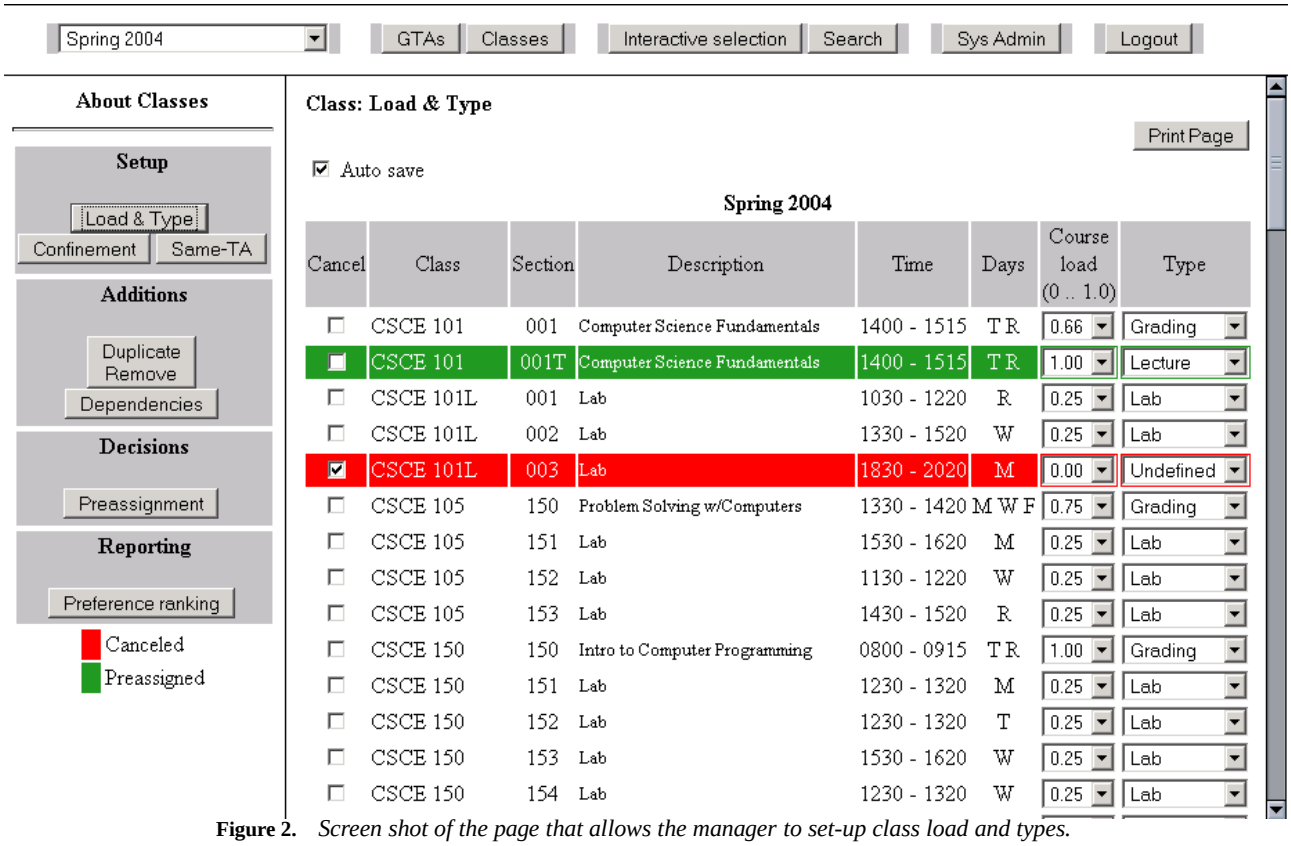


**Figure 2.** *Screen shot of the page that allows the manager to set-up class load and types.*

preferences of the GTAs. Because the hard constraints cannot be broken, the problem cannot be modeled as a MAX-CSP [4]. We choose to express the courses as the variables of a Constraint Satisfaction Problem (CSP) and the GTAs as the values that these courses may take. We express the constraints that restrict the acceptable assignments as unary, binary, and non-binary constraints. Our goal is thus to find the longest partial and consistent solution while maximizing the GTA preferences. We tested various optimization criteria, such as maximizing the minimal preference in a solution, maximizing the average or geometric mean of preference values, etc. Details of the modeling can be found in [5, 8]. We also proposed to reformulate some of the non-binary constraints into binary ones and evaluated the benefits of this reformulation [6, 7].

### 2.4.2 *Interactive decision making*

The manager can adopt one of two dual perspectives on the problem and switch between them: assigning GTAs for courses or vice versa. Fig. 3 shows how the appropriate GTAs (i.e., those that have passed node consistency) are listed to the user. The upper portion of the pull-down displays the GTAs that can be assigned to the course without reservation. In terms of the CSP, these are the values in the current domain of the variable. The lower portion lists the GTAs who could potentially be assigned but are 'busy' in assignments and those who can be relieved. These are the values eliminated from the initial domain of the variable by constraint propagation. In each portion, and the GTAs are listed in decreasing preference order, as a primary criterion, then in increasing lexicographical of their last name, as a secondary criterion. Next to the name, the 'current' capacity of each GTA is displayed (which is the hired capacity of the GTA discounted by the load of his/her other assignments). Every time a new assignment is made, a full arc-consistency algorithm is executed to filter

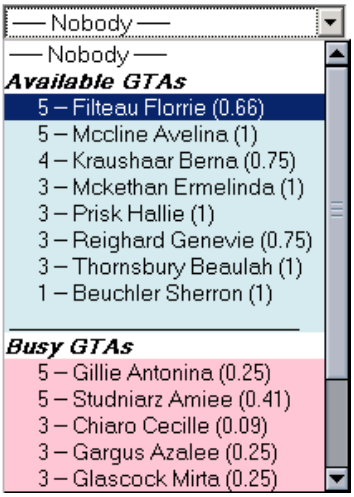


**Figure 3.** *Checking available/busy GTAs for a course.*

the domains of the unassigned variables. Assignments can also be undone or changed. The domains of all the variables are then recomputed from scratch while maintaining the intermediate selections. We decided against using a dynamic arc-consistency algorithm [2] to avoid the use of additional data structures and because the solution adopted is quite efficient in practice. Naturally, this decision may need to be reconsidered if the application size or characteristics are modified. This functionality will be described in detail in [14].

### 2.4.3 *Automated search algorithms*

We have developed and tested various search strategies, which we summarize below. Using the 8 data-sets (pertaining to different aca-

demic semesters) of the same problem and comparing the performance of search strategies for these problems was particularly instructive as it allowed us to recognize known features and identify new characterizations of these search procedures, shown in Tab. 1.

**Heuristic backtrack search:** Because the problem is often over-constrained, we changed the backtrack mechanism of depth-first search with forward-checking to force search to pursue a partial solution even when the domains of some future variables are wiped out. We tested various ordering heuristics for variables and values and found that both the dynamic variable ordering with the least domain heuristic and the minimum ratio of domain size to degree yielded the best results. Although BT is theoretically sound and complete, the size of the search space (around 60 variables and 30 values) makes such guarantees meaningless in practice. Fig. 4 illustrates thrashing, where search never recovers from early bad decisions. Indeed, the shallowest level of backtrack achieved after 24 hours (26%) is not significantly better than that reached after 1 minute (20%) of search. More details can be found in[5, 8, 7, 10].

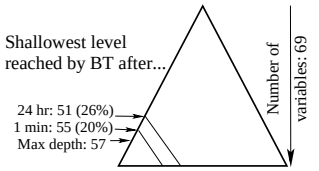


**Figure 4.** *BT search thrashing in large search spaces.*

**Stochastic local search:** Our local search (LS) is a hill-climbing search using the min-conflict heuristic for value selection [13]. It is documented in [17, 16]. It begins with a complete assignment (not necessarily consistent) and tries to improve it by changing inconsistent assignments in order to reduce the number of constraint violations. We propagate the effect of consistent assignments over the domains of the variables with inconsistent assignments. This design decision allows us to effectively handle non-binary constraints. We implement local search in a greedy fashion in the sense that we do not backtrack over consistent assignments. Moreover, we apply a random-walk strategy to escape from local optima [1]. With a probability $(1-p)$, we choose the value of a variable using the min-conflict heuristic, and with probability $p$ we choose this value randomly. Following the indications of [1], we choose $p = 0.02$. Further studies on the choice of the value of $p$ for the GTA problem are reported in [16]. Furthermore, we use random restarts to break out of local optima.

**Multi-agent search (ERA):** Liu et al. [12] proposed the ERA (Environment, Reactive rules, and Agents) algorithm, a multi-agent-based search for solving CSPs. Each variable is an agent. The position of an agent corresponds to the value assigned to this variable. The environment records the number of constraint violations of each agent's position. An agent moves according to its reactive rules to occupy a position where no constraint is broken. A solution is found when all agents are in such positions. On the surface, this technique appears to be a variation of local search. This algorithm acts as an 'extremely' decentralized local search, where any agent can move to any position, possibly forcing other agents to seek other positions. Our experiments uncovered the ERA's immunity to local optima [17, 16]: it is indeed the only technique that consistently solved all tight problems. We also uncovered the weakness of ERA on over-constrained problems, where a deadlock phenomenon undermines its stability resulting in particularly short solutions. However, the deadlock phenomenon is useful to identify conflicts in a compact manner.

**Randomized backtrack search:** Gomes et al. [9] proposed to avoid the thrashing of BT by introducing randomness in the variable selection heuristic and using a restart strategy. This allows search to explore wider areas of the search space. Using a random ordering of variable-value pairs, we implemented the Randomization and Geometric Restarts (RGR) strategy of Walsh [15]. According to RGR, search proceeds until it reaches a cutoff value for the number of nodes visited. The cutoff value for each restart is a constant factor that is larger than the previous run. The initial cutoff is equal to the number of variables. We have introduced a simple by effective improvement to RGR: Randomization and *Dynamic* Geometric Restarts (RDGR) [10, 11]. It operates by not increasing the cutoff value for the following restart whenever the quality of the current best solution is not improved upon. When the current restart improves on the current best solution, then the cutoff value is increased geometrically, similar to RGR. Because the cutoff value does not necessarily increase, completeness is no longer guaranteed. This situation is acceptable in application domains (like ours) with large problem size where completeness is, anyway, infeasible in practice. Smaller cutoff values result in a larger number of restarts taking place in RDGR than RGR, which increases the probability of finding a solution. We have compared the performance of RDGR to that of RGR on the data collected in our project over a wide range of running times and using the cumulative distribution and probability density functions of the solutions. Our investigations have shown that RDGR clearly dominates RGR on the GTA assignment problem. We are currently testing it on randomly generated CSPs.

## 2.5 CHARACTERIZATION & PRACTICAL USE

We noticed that when the result of an algorithm (e.g., ERA) is a used as a seed to another algorithm (e.g., BT), the resulting solution has better quality than the two algorithms used independently. We are now studying how to integrate the various search mechanisms in a collaborative manner by exploiting and extending the knowledge about the behavior of these algorithms summarized in Tab. 1 (e.g., completeness, soundness, response time, characteristic performance on solvable and over-constrained problems, stability of solutions, and quality of solutions).

| **General Observations** |
|---|
| **ERA:** Stochastic and incomplete. |
| **LS:** Stochastic, incomplete, and quickly stabilizes. |
| **RDGR:** Stochastic, incomplete, immune to thrashing, produces longer solutions than BT, immune to deadlock, reliable on instances & immune to local optima, but less than ERA. |
| **RGR:** Stochastic, approximately complete, less immune to thrashing than RDGR & yields shorter solutions than RDGR in general. |
| **BT:** Systematic, complete (theoretically, rarely in practice), liable to thrashing, yields shorter solutions than RDGR and RGR, stable behavior & more stable solutions than stochastic methods in general. |

| **Tight but solvable problems** |
|---|
| **ERA:** Immune to local optima, solves tight CSPs, gives best results. |
| **LS:** Liable to local optima, fails to solve tight CSPs even with random-walk & restart strategies. |
| **RDGR, RGR:** RDGR clearly dominates RGR, but is less good than ERA |

| **Over-constrained problems** |
|---|
| **ERA:** Deadlock causes instability and yields shorter solutions. |
| **LS, RGR, RDGR, BT:** Finds longer solutions than ERA. |
| **RDGR:** Gives best results. |

**Table 1.** *Empirical characterization on the GTA problem.*

Currently, the user decides which search mechanism to execute, the search processes running independently. On the collected data sets, the response time of interactive search is immediate for the human user, but that of automated search is not. Our tests showed that a running time from 5 to 10 minutes yields as good solutions as can be expected from all search algorithms. Although some of the search algorithms are theoretically complete, the quality of the solutions do not appreciably improve with time. Since we started using the system in August 2001, the manager typically generates first solutions with each of the automated searches, then uses these solutions as drafts for building up the final solution with interactive search.

## 3 BENEFITS & IMPACT

The task of assigning GTAs to tasks is particularly difficult for humans because of all of the hard constraints about students' availability, qualification, and load that one has to keep track of. In our department, it is acknowledged to be the 'most difficult duty of a department chair.' On the other hand, the manager has plenty of information about the GTAs, the courses, and the teaching faculty at his/her fingertips. It is illusive to attempt to formally model such constraints because they are typically subjective and infinite. Also, they would complicate the problem solving without necessarily improving the quality of the solutions. *Fortunately in our particular setting, computers seem to be most effective in exactly the same tasks where humans fail* (e.g., keeping track of hard constraints), *and vice versa* (e.g., quickly evaluating alternatives and making compromises) The fact that our system keeps the human user in the decision loop while removing the need for tedious (and error-prone) manual hand assignments explains the success of the system in our setting.

Initially, the entire process was carried out manually and on paper. The system, with barely a few functionalities, was deployed within a few months. Functionalities were added as they were identified, often in a need-driven manner and in close collaboration between the designers of the system and its users. However, the positive effects of the system on the GTA assignment task were immediately noticeable. Indeed, our endeavor has allowed the department to gradually redesign its processes for the data acquisition mechanisms, the GTA selection and hiring procedures, and the assignment policies. The department is currently working towards standardizing and streamlining the entire process.

Ours was an ad-hoc approach to system development, and we did not carry out a formal longitudinal study of usability and usefulness. However, anecdotal evidence of the value of the system is demonstrated in the form of continued financial support for one student, overjoyed satisfaction of the staff (relieved from handling application forms and massive paperwork), and enthusiastic anonymous online reviews from the students. Invariable, assignments are now made quickly (from a 3-week duration down to a day or two), they are more stable, and, above all, they are more satisfactory to the course instructor, the GTAs, and the students in the classrooms.

## 4 FUTURE WORK

Although we already have a working system, we are far from completing the project and implementing the ideas that are continuously popping up. We constantly review the functionalities and the performance of our prototype and adapt it to changing department needs. The main areas we plan to address next are the following: designing hybrid search procedures that work seamlessly together, enabling the manager to compose interactively partial solutions built manually or with search, allowing the manager to participate actively in the search process and monitoring and guiding its progress, and finally developing visualization widgets able to display pertinent statistics about a problem as solutions are being built.

## 5 CONCLUSION

We have developed an interactive system for hiring and managing graduate teaching assistants. Our prototype system has integrated interactive-selection and search capabilities, thus alleviating the burden of the manager and allowing the quick development of stable solutions. This system has also paved avenues for research work in Constraint Processing and other related areas.

## ACKNOWLEDGEMENTS

The development of the database and web interfaces is supported by the Department of Computer Science & Engineering of the University of Nebraska-Lincoln. The development of the algorithms is supported by grant #EPS-0091900 of the National Science Foundation. Several staff members and students of our department, too many to list here, contributed to the progress and success of this application by providing domain expertise and practical feedback.

## REFERENCES

[1] R. Barták. On-Line Guide to Constraint Programming, 1998.
[2] C. Bessière, 'Arc-Consistency in Dynamic Constraint Satisfaction Problems', in *Proc. of AAAI 91*, pp. 221–226, (1991).
[3] R. Dechter. Home web page, 2001.
[4] E.C. Freuder and R.J. Wallace, 'Partial Constraint Satisfaction', *Artificial Intelligence*, **58**, 21–70, (1992).
[5] R. Glaubius. A Constraint Processing Approach to Assigning Graduate Teaching Assistants to Courses. Undergraduate Honors Thesis. CSE-UNL, 2001.
[6] R. Glaubius and B.Y. Choueiry, 'Constraint Modeling and Reformulation in the Context of Academic Task Assignment', in *Workshop on Modelling and Solving Problems with Constraints (ECAI 02)*, (2002).
[7] R. Glaubius and B.Y. Choueiry. Constraint Modeling and Reformulation in the Context of Academic Task Assignment. Poster presentation at SARA 02, 2002.
[8] R. Glaubius and B.Y. Choueiry, 'Constraint Modeling in the Context of Academic Task Assignment', in *Proc. of CP'02*, LNCS 2470, p. 789, (2002).
[9] C.P. Gomes, B. Selman, and H. Kautz, 'Boosting Combinatorial Search Through Randomization', in *Proc. of AAAI 98*, pp. 431–437, (1998).
[10] V. Guddeti, *Empirical Evaluation of Heuristic and Randomized Backtrack Search*, Master's thesis, CSE-UNL, 2004. Forthcoming.
[11] V. Guddeti, H. Zou, and B.Y. Choueiry. An Empirical Study of Heuristic and Randomized Search Techniques in a Real-World Setting, 2004. Under review.
[12] J. Liu, H. Jing, and Y.Y. Tang, 'Multi-agent oriented constraint satisfaction', *Artificial Intelligence*, **136**, 101–144, (2002).
[13] S. Minton, M. D. Johnston, A. B. Philips, and P. Laird, 'Minimizing Conflicts: A Heuristic Repair Method for Constraint Satisfaction and Scheduling Problems', *Artificial Intelligence*, **58**, 161–205, (1992).
[14] V.R. Thota. Using Constraint Propagation to Support Interactive Resource Allocation. Masters Project, CSE-UNL, 2004. Forthcoming.
[15] T. Walsh, 'Search in a Small World', in *Proc. of IJCAI 99*, pp. 1172–1177, (1999).
[16] H. Zou, *Iterative Improvement Techniques for Solving Tight Constraint Satisfaction Problems*, Master's thesis, CSE-UNL, 2003.
[17] H. Zou and B.Y. Choueiry, '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 (CP 02)*, pp. 81–101, (2003).