AI 2006

Richard Ellis,

Tony Allen

and Andrew Tuson (Eds)

APPLICATIONS AND INNOVATIONS IN INTELLIGENT SYSTEMS XIV

Proceedings of AI-2006, the Twenty-sixth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence

Springer

Applications and Innovations in Intelligent Systems XIV

Richard Ellis, Tony Allen and
Andrew Tuson (Eds)

Applications and Innovations in Intelligent Systems XIV

Proceedings of AI-2006, the Twenty-sixth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence

Springer

Richard Ellis, BSc, MSc
Stratum Management Ltd, UK

Dr Tony Allen, PhD
Nottingham Trent University, UK

Dr Andrew Tuson, MA, MSc, PhD, MBCS
Department of Computing, City University, London

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

ISBN-10: 1-84628-665-4
ISBN-13: 978-1-84628-665-0

Printed on acid-free paper

Printed in the United Kingdom

9 8 7 6 5 4 3 2 1

Springer Science+Business Media
springer.com

APPLICATION PROGRAMME CHAIR'S INTRODUCTION

RICHARD ELLIS
Managing Director, Stratum Management Ltd, UK

The papers in this volume are the refereed application papers presented at AI-2006, the Twenty-sixth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, held in Cambridge in December 2006. The conference was organised by SGAI, the British Computer Society Specialist Group on Artificial Intelligence.

This volume contains seventeen refereed papers which present the innovative application of a range of AI techniques in a number of subject domains. This year, the papers are divided into sections on *Data Mining and Bayesian Networks; Genetic Algorithms and Optimisation Techniques; Agents and Semantic Web; and Natural Language.*

This year's Rob Milne Memorial Award for the best refereed application paper was won by a paper entitled "Managing Restaurant Tables using Constraints". The authors are Alfio Vidotto and Kenneth N. Brown (University College Cork, Ireland) and J. Christopher Beck (University of Toronto, Canada). This award was instituted in 2005 in memory of the contribution that the late Rob Milne made to AI.

This is the fourteenth volume in the *Applications and Innovations* series. The Technical Stream papers are published as a companion volume under the title *Research and Development in Intelligent Systems XXIII.*

On behalf of the conference organising committee I should like to thank all those who contributed to the organisation of this year's application programme, in particular the programme committee members, the executive programme committee and our administrator Mark Firman.

Richard Ellis
Application Programme Chair, AI-2006

ACKNOWLEDGEMENTS

AI-2006 CONFERENCE COMMITTEE

Dr. Andrew Tuson, City University	(Conference Chair)
Dr. Tony Allen, Nottingham Trent University	(Past Conference Chair and Deputy Application Program Chair)
Dr. Alun Preece, University of Aberdeen	(Deputy Conference Chair, Electronic Services)
Dr Frans Coenen, University of Liverpool	(Deputy Conference Chair, Local Arrangements and Deputy Technical Programme Chair)
Prof. Adrian Hopgood, Nottingham Trent University	(Workshop Organiser)
Rosemary Gilligan, University of Hertfordshire	(Treasurer)
Dr Nirmalie Wiratunga, The Robert Gordon University, Aberdeen	(Poster Session Organiser)
Richard Ellis, Stratum Management Ltd	(Application Programme Chair)
Professor Max Bramer, University of Portsmouth	(Technical Programme Chair)
Alice Kerly, University of Birmingham	(Research Student Liaison)
Dr. Miltos Petridis, University of Greenwich	(UK CBR Organiser)
Mark Firman, City University	(Conference Administrator)

APPLICATION EXECUTIVE PROGRAMME COMMITTEE

Richard Ellis, Stratum Management Ltd (Chair)
Dr Tony Allen, Nottingham Trent University (Deputy Chair)
Rosemary Gilligan, University of Hertfordshire
Prof. Adrian Hopgood, Nottingham Trent University
Dr Miltos Petridis, University of Greenwich
Mr. Richard Wheeler, University of Edinburgh

APPLICATION PROGRAMME COMMITTEE

CONTENTS

SESSION 3: AGENTS AND SEMANTIC WEB

SESSION 5: NATURAL LANGUAGE

SHORT PAPERS

BEST APPLICATION PAPER

Managing Restaurant Tables using Constraints

Alfio Vidotto[1], Kenneth N. Brown[1], J. Christopher Beck[2]

[1] Cork Constraint Computation Centre, Department of Computer Science, University College Cork, Ireland.
av1@student.cs.ucc.ie, k.brown@cs.ucc.ie

[2] Toronto Intelligent Decision Engineering Laboratory, Department of Mechanical and Industrial Engineering, University of Toronto, Canada.
jcb@mie.utoronto.ca

Abstract

Restaurant table management can have significant impact on both profitability and the customer experience. The core of the issue is a complex dynamic combinatorial problem. We show how to model the problem as constraint satisfaction, with extensions which generate flexible seating plans and which maintain stability when changes occur. We describe an implemented system which provides advice to users in real time. The system is currently being evaluated in a restaurant environment.

1. Introduction

Effective table management can be crucial to a restaurant's profitability – inefficient use of tables means that the restaurant is losing potential custom, but overbooking means that customers are delayed or feel cramped and pressured, and so are unlikely to return. In addition, customer behaviour is uncertain, and so seating plans should be flexible or quickly reconfigurable, to avoid delays. The restaurant manager is faced with a series of questions. Should a party of two be offered the last four-seater table? For how long should we keep a favourite table for a regular customer? Should a party of four be offered a table for 8 p.m.? If no table is available at 7 p.m., what other times should be offered? When a party takes longer than expected, can we re-assign all diners who have not yet been seated to avoid delays? When a party doesn't appear, can we re-assign all other diners to gain an extra seating? In Computer Science terms, table management is an online constrained combinatorial optimisation problem – the restaurant must manage reservations, and manage unexpected events in real-time, while maximising the use of its resources.

In this paper, we describe an implemented solution to the restaurant table management problem which helps managers answer the above questions. The solution is based on constraint programming, and handles both flexibility and stability. The system we describe is currently being evaluated in a restaurant. The remainder of the paper is organised as follows. Section 2 presents more details of the table management problem, and describes one particular restaurant. Section 3 reviews the necessary elements of constraint programming. Section 4 presents a basic constraint model and search algorithm. Section 5 extends the model to represent flexibility, and to search for flexible plans, while section 6 describes our approach to finding stable plans. Section 7 presents the user interface for our implemented system. Finally, section 8 describes conclusions and future work.

2. Restaurant Table Management

Eco [1] is a popular medium-size restaurant in Douglas, Cork City, with a high turnover seven days a week. It was a pioneer in computer and internet solutions, first offering email booking in 2000. The restaurant has 23 tables, ranging in size from 2 to 8 (Figure 1). Some of the table capacities depend on the state of other tables: for example, tables 2 and 15 can both seat 6, but when one is occupied by 5 or 6 diners, then the other can seat at most 4. The tables can also be reconfigured: for example, the 2-seater tables 21 and 22 can be joined to accommodate 3 to 5 diners. The maximum party size that can be seated at a conjoined table is 30. There are over 100 different possible restaurant configurations, and thus the restaurant capacity ranges from 85 to 94. An evening session in the restaurant begins at 4 p.m., and the last party should be seated by 10:30 p.m. As a guide, the restaurant aims to have between 190 and 210 covers (individual diners) each evening – fewer than that, and the tables are not being well utilised; more than that, and the kitchen will be stretched to provide the food on time. Table management in Eco, as in most restaurants, has two distinct phases: *booking* and *floor management.*

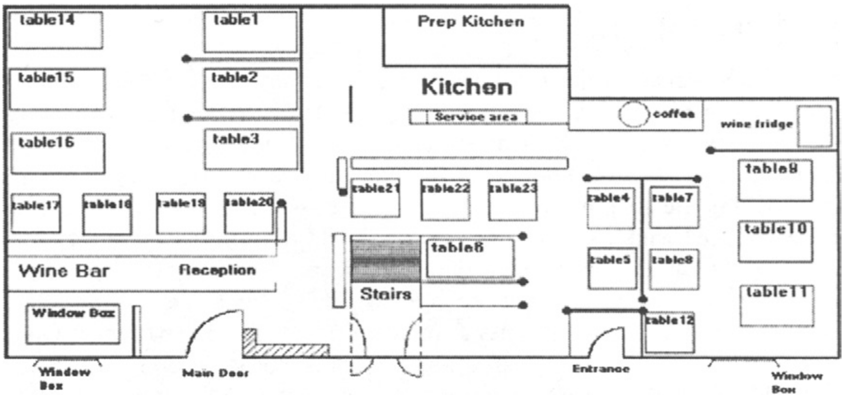

Figure 1: Layout of the restaurant *Eco*

In the booking phase, the booker must negotiate start times with customers to ensure that customers' requirements are satisfied, while maintaining a flexible table assignment that maximises the chances of being able to seat the desired number of covers. Typically, the booker will allocate specific tables to each booking request, and these rarely change; when a request cannot be accommodated on the current booking sheet, either the customer must be persuaded to accept another time, or the request must be declined. It is possible, however, that a reallocation of diners to tables would allow the new request to be accepted. In some cases, in order to maintain a balanced plan, a restaurant will decline a booking, or suggest a different time, even if a table is available. In addition, the booker must estimate the expected duration of the meal, based on the characteristics of the booking (including time, day of the week, and party size).

In floor management, the objectives are different. The evening starts with a partially completed booking sheet. The customers have been given definite times, and the aim is now to seat the customers with minimum delay, to modify the seating plan when changes happen, and to accept or decline "walk-ins" – customers arriving at the restaurant without a booking. The main challenge is that individual customers are unpredictable – they may arrive late, they may not arrive at all, they may take longer or shorter than expected, they may change the size of their party, and they may arrive believing a booking has been made when none has been recorded. The floor manager must make instant decisions, balancing current customer satisfaction with expectations for the rest of the evening.

The initial problem is to construct an interactive software tool, which assists restaurant staff in both the booking and floor management phases. As a research problem, our goal is to evaluate whether constraint programming techniques can provide support for the dynamic and uncertain aspects of the problem. If the research prototype is successful, a new tool will be developed, and incorporated into customer relationship management software.

3. Constraint Programming

A *Constraint Satisfaction Problem* (CSP) is defined by a set of decision variables, $\{X_1, X_2, \ldots, X_n\}$, with corresponding domains of values $\{D_1, D_2, \ldots, D_n\}$, and a set of constraints, $\{C_1, C_2, \ldots, C_m\}$. Each constraint is defined by a scope, i.e. a subset of the variables, and a relation which defines the allowed tuples of values for the scope. A state is an assignment of values to some or all of the variables, $\{X_i=v_i, X_j=v_j, \ldots\}$. A solution to a CSP is a complete and consistent assignment, i.e. an assignment of values to all of the variables, $\{X_1=v_1, X_2=v_2, \ldots, X_n=v_n\}$, that satisfies all the constraints. The standard methods for solving CSPs are based on backtracking search interleaved with constraint propagation. An introduction to constraint programming can be found in [2], while [3] surveys recent research.

For search, the order in which variables and values are tried has to be specified as part of the search algorithm, and has a significant effect on the size of the search tree. The standard variable ordering heuristic chooses the variable with the smallest current domain, or the smallest ratio of domain size to the number of constraints acting on the variable. For an instance of a CSP, a single run with a single ordering heuristic can get trapped in the wrong area of the search tree. To avoid this, randomized restarts have been proposed [4] – for a single heuristic, if no result has been found by a given time limit, the search is started again. Tie breaking and value ordering are done randomly, and so each restart explores a different path. Similarly, algorithm portfolios [5] interleave a set of randomized algorithms. In [6] search robustness is enhanced by combining multiple variable and value ordering heuristics with time-bounded restarts.

In constraint propagation, the domains of unassigned variables are reduced by removing values which cannot appear in any solution that extends the current state. For example, if we have the constraint X < Y, and X and Y's domains are {2,3,4,5} and {1,2,3,4} respectively, then the values 4 and 5 can be removed from X's domain, and 1 and 2 from Y's domain, since none of those values could possibly satisfy the constraint. Reducing the domains reduces the size of sub-tree that has to be explored. A large part of the success of constraint programming tools is due to efficient domain filtering algorithms for specialised constraints; e.g. [7].

Dynamic problems are problems that change as the solution is being executed – for example, in scheduling, a machine may break down, or a scheduled action may be delayed due to the late arrival of supplies. Dynamic CSPs [8] model changes to problems. The aim may be to minimise the effort to find new solutions, or to minimise the distance between successive solutions. Attention has recently turned to problems where we have some model of what the changes might be. Both [9] and [10] reason about the probability of future events: [9] searches and propagates constraints over a tree of possible futures; [10] samples possible futures, and then selects an action which minimises regret over the samples. [11] searches for optimally stable solutions. They start with the original solution and iteratively check whether reassigning one variable, two variables, etc., is sufficient to solve the new problem. [12] proposes special stability constraints. Some approaches aim to prevent instability by providing robust solutions. In [13] flexible solutions to scheduling problems are achieved by adding slack to activity durations. Super solutions [14] are solutions that guarantee a limited number of repairs in case of changes.

4. Modelling the static table management problem

As discussed in section 2, the restaurant problem is inherently dynamic, but we can view it a sequence of static problems, each linked by a set of changes. In this section, we describe our representation of the static problem as a CSP, and discuss our algorithm for solving it.

We model table management as a scheduling problem, viewing tables as resources, and parties as tasks. Each party has a fixed start and end time, and a size. Each party must then be allocated to a table (or set of tables), such that the table is large enough for the party, and such that no two parties that overlap in time are allocated to the same table. Each party must be seated without interruption on the table. The problem is to determine whether or not a set of parties can be seated, and to provide a feasible seating plan if there is one. Despite having fixed start and end times, the underlying scheduling problem is NP-complete [15]. Figure 2 shows a problem instance with five parties (left) and a possible allocation (right), where tables T_2 and T_3 have been joined for the first two time slots.

To represent this as a CSP, we model the parties as decision variables, and the tables as the values to be assigned. The detailed constraint model is generated automatically from a template and from details of the restaurant. Figure 3 shows the resulting model for the simple problem of Figure 2. The variables P_1, P_2, P_3, P_4, and