12th International Conference on Industrial and Engineering Applications of AI and Expert Systems, IEA/AIE-99, Cairo, Egypt, 1999

The Design of a Multi-Tiered Bus Timetabling System

Hon Wai Chun[1] and Steve Ho Chuen Chan[2]

[1] City University of Hong Kong, Department of Electronic Engineering
Tat Chee Avenue, Kowloon, Hong Kong
eehwchun@cityu.edu.hk
[2] Advanced Object Technologies Ltd, Unit 602A HK Industrial Technology Centre
72 Tat Chee Avenue, Kowloon, Hong Kong
steve@aotl.com

Abstract. This paper describes the design of the Bus Timetabling System (BTS) we have developed for one of the largest privately held bus companies in the world. This Bus Company operates close to 3,500 buses and employs over 7,500 bus drivers. The BTS generates a timetable for each bus route using "distributed constraint-based search" (DCBS). The DCBS algorithm combines techniques of constraint programming, heuristic search, with distributed scheduling. This allows multiple types of constraints to be considered, ensures that idle resources can be shared, and yet generates a timetable within reasonable time. The Bus Timetabling System also determines the bus captain duty assignments for the scheduled bus routes. This paper focuses on both the AI methodologies used and the design of the client-server multi-tiered architecture. We used a distributed object architecture to implement our distributed scheduling system.

1 Introduction

This paper describes the design of a practical application that combines state-of-the-art distributed AI methodologies with state-of-the-art distributed software architecture to perform bus timetabling. The Bus Timetabling System (BTS) uses a novel AI algorithm, which we call Distributed Constraint-based Search (DCBS) [3], that combines distributed [4, 10] and constraint-based [13, 14] scheduling with heuristic search. The DCBS algorithm automatically generates a bus timetable based on a set of service frequency requirements, duty assignment constraints, operational nature of the route, and the type of buses available. Even for a single bus route, the timetabling search space is quite large and highly non-linear; a traditional constraint-satisfaction problem (CSP) algorithm will not suffice [6, 9]. The DCBS algorithm solves this problem by distributing the scheduling responsible among a collection of scheduling agents.

The Bus Company we developed BTS for is quite unique in its operations. In other companies, bus timetabling and duty assignment are considered as independ-

ent scheduling tasks, or at least to have minimal dependencies. For our Bus Company, each bus is associated with a driver. If a driver takes a rest or meal break, the bus becomes idle. Therefore, duty assignment constraints, such as duty duration and meal requirements, must also be considered at the same time with the bus timetabling constraints such as service frequency requirements. BTS uses constraint-programming techniques to encode these different types of constraints.

To implement the distributed scheduling algorithm, we used distributed object computing (DOC) techniques. The architecture is basically a multi-tiered client-server architecture where the application server consists of a collection of distributed scheduling agents or processes. Each of these distributed scheduling process contains distributed objects, represented as DCOM components, that capture knowledge and constraints related to a bus route.

This paper first describes the distributed AI methodology used. It then describes the design of a distributed software architecture that implements this methodology.

2 The DCBS Algorithm

Figure 1 shows the overall structure of the DCBS bus-timetabling algorithm. The distributed scheduling system consists of a shared Timetabling Blackboard and a collection of Distributed Scheduling Agents. Each scheduling agent contains a DCBS algorithm to perform timetabling. There is one distributed scheduling agent per bus route to be scheduled.

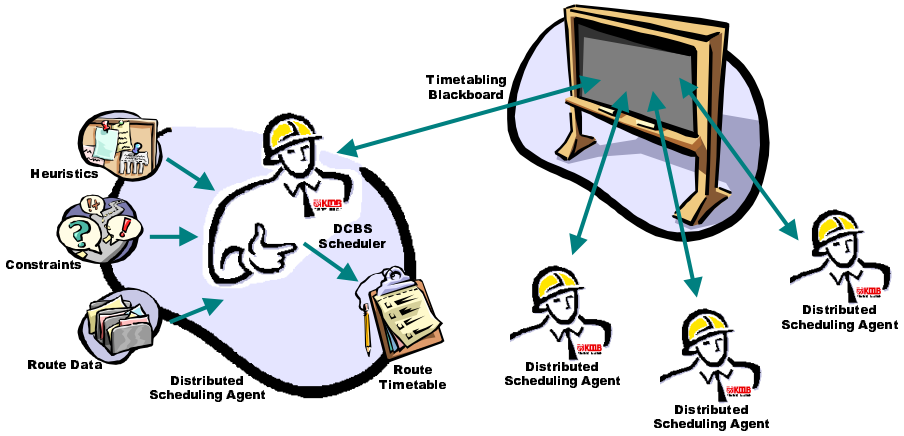

Fig. 1 The structure of the DCBS bus timetabling algorithm.

The timetabling blackboard is the main communication channel between all the distributed scheduling agents and contains service requests that cannot be fulfilled by a route's own set of buses. Distributed agents with idle resources will check the timetabling blackboard for additional work assignment.

Each distributed scheduling agent contains only data for the route it is responsible for scheduling. There are four types of route data. The first is general route

parameters, such as the number of available buses, number of runs, route start and end times, terminuses to be served, etc. The second is a table of the journey times between terminuses of routes at different times of the day and day of the week. The third is the desired headway table for a route. The fourth is the operational parameters; such as the minimum rest/meal break duration, time periods when meal breaks should be taken, etc.

In addition, each distributed scheduling agent uses two types of scheduling knowledge to generate a route timetable – a set of timetabling constraints and a set of heuristics to help guide the scheduling search. Although each agent only generates a timetable for its own route, the timetabling blackboard allows buses and bus driver resources to be shared among different routes.

2.1 Bus Timetabling Constraints

BTS uses object-oriented constraint-programming techniques [8, 12] to encode the business logic or operational constraints. The following lists some of the key bus timetabling constraints that are considered by BTS:

- **Journey Time**. The first priority is to ensure that there is enough travelling time between terminuses while making all the necessary stops in between.
- **Layover Breaks**. Bus drivers are given a short *layover break* at the end of each journey; duration is roughly 10% of the journey time.
- **Meal Breaks**. Bus drivers must be assigned a *meal break*. For routes with uniform service frequencies, relief drivers are scheduled.
- **Service Frequencies**. The *service frequency*, or *headway*, defines how often buses leave each terminus at different times of the day.
- **Light Runs**. When there are not enough buses, additional buses may need to travel directly from the other terminus without passengers.
- **Inter-Workings**. When there are more buses than needed, these buses may be scheduled to service one or more other routes.
- **Home Depot**. Buses have assigned home depots where they are scheduled to return for storage overnight.
- **Supplementary Buses**. Sometimes *supplementary buses* will be used to service morning/afternoon peak hours. These drivers must work in *split-shift* duties.

2.2 Variables to be Scheduled

During the process of producing a bus timetable, the DCBS scheduling algorithm must schedule or assign values to many types of timetable unknowns. Some of

these unknowns are encoded as constraint-satisfaction-problem (CSP) constrained variables.

- **Departure Time**. This depends on service frequency requirements. The exact service frequency will also depend on the bus's departure time.
- **Bus Sequence**. At any time, there may be several buses at a terminus. The scheduling algorithm must select a bus to depart next.
- **Journey Time**. The journey time will depend on the route, direction of travel and the time of the day.
- **Rest/Meal Break**. At the end of each journey, the scheduling algorithm must decide whether to assign a rest or meal break if parking is available.
- **Rest/Meal Break Duration**. The duration of the break should be assigned opportunistically to match headway requirements.
- **Light Runs**. The scheduling algorithm must decide when to schedule light runs or use supplementary buses.
- **Inter-Workings**. The scheduling algorithm must also decide when it is appropriate to free a bus for an inter-working run.

2.3 Scheduling Criteria

The DCBS timetabling algorithm was designed with several scheduling criteria and objectives in mind. These objectives are user adjustable such that each route may have a different set of scheduling objectives.

- **Minimise Total Buses Needed.** The scheduling algorithm should only use just enough buses to satisfy the service frequencies.
- **Minimise Supplementary Buses Needed**. The scheduling algorithm should minimise the use of supplementary buses if possible.
- **Minimise Fluctuation in Headway.** The time between each departure from a terminus must be as close to the frequency requirements as possible.
- **Optimise Resource Usage**. The scheduling algorithm should minimise idle time. Buses with long idle time are scheduled to service other routes.

In terms of AI methodologies, the DCBS algorithm we developed for bus timetabling uses distributed scheduling as a framework to reduce search space and yet provide a mean of co-operation between scheduling agents. It uses constraint programming to encode inter-related bus timetabling and duty assignment constraints, and heuristic search to improve overall timetabling performance. The following Section explains how the DCBS algorithm is actually implemented and the software

technologies we used.

3 The Software Architecture

In terms of the implementation software architecture, the BTS has a multi-tiered client-server architecture. The system consists of three key software modules: the BTS Client Program, the BTS Scheduling Program, and the BTS Database Program.

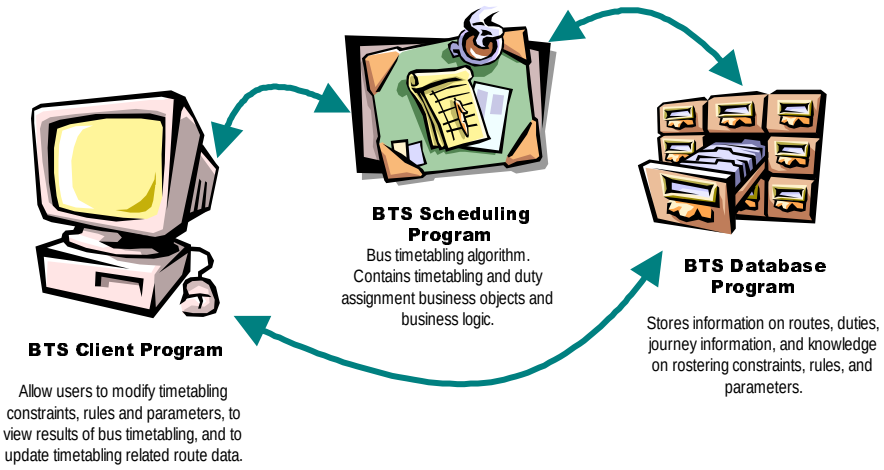

Fig. 2 The BTS multi-tiered software architecture.

- **Client-Tier**. Each BTS workstation contains a copy of the "BTS Client Program." The BTS Client Program is built using Microsoft Visual Basic and contains user-friendly menus and screens that allow users to input and modify bus timetabling and duty assignment constraints, route information, headway tables, journey times, etc. Through the BTS Client Program, users view and edit BTS-generated bus timetables and duty assignments. The Client Program also allows the user to print operational and management reports.
- **Server-Tier**. The "BTS Scheduling Program" is an application server built using distributed object technology and contains a collection of DCBS scheduling agents. Each scheduling agent contains all the necessary business objects and business logic needed to generate a bus timetable and duty assignment of a single route. The BTS Scheduling Program is installed only on the BTS server machine. In other words, the BTS server machine will be responsible for generating the bus timetables for all the remote BTS client workstations. The scheduling agents automatically load appropriate constraints, rules, and parameters from the BTS database. Once a bus

timetable is generated, it is stored back into the BTS database for later retrieval, update, or modification.

- **Database-Tier**. The third software component is the "BTS Database Program." The Database Program also resides on the BTS server machine and coordinates all data requests from all the remote BTS client machines. It supports a set of system administrative functions, such as database backup and restore, database maintenance, user security, etc.

3.1 Software Operation

Operating BTS is relatively straightforward. There are only three main steps:

- **Step 1.** The first step in generating a bus timetable and duty assignment for a particular bus route is to enter all information related to that route, such as headway frequencies, number of buses, terminuses, etc., into the BTS. In addition, the user can update any timetabling data that might have changed, such as journey times, operational requirements, etc. The user can also add, modify, or remove any constraints, rules, and parameters related to bus timetabling.
- **Step 2.** Once all necessary route and timetabling information have been entered and updated, the user can then generate a new bus timetable. Once a bus timetable has been generated, it is displayed to the user in the form of a user-friendly spreadsheet. The user can generate as many timetables for a route as needed with different sets of constraints and parameters (in "what-if" situations). The BTS user can interactively refine a bus timetable by making manual modifications and freezing portions of the timetable before requesting the BTS to produce a new version.
- **Step 3.** Once the bus timetable is finalised, the user can print operational and management reports and produce export files to transfer to the corporate mainframe machines.

4 The Distributed Environment

To encode the distributed AI methodologies in BTS, we decided to use distributed object computing (DOC) technologies. For distributed scheduling, each DCBS scheduling agent is encoded as a DCOM distributed object. C++ is used for implementation. To encode constraints, we used a commercial object-oriented constraint-programming class library [5, 11, 12]. This class library also provides hooks to encode heuristics that are used during constraint-based search. On top of the library is a scheduling framework [1, 2] that is designed to solve a general class of resource

allocation problems. The DCBS timetabling blackboard is implemented as relational database tables.

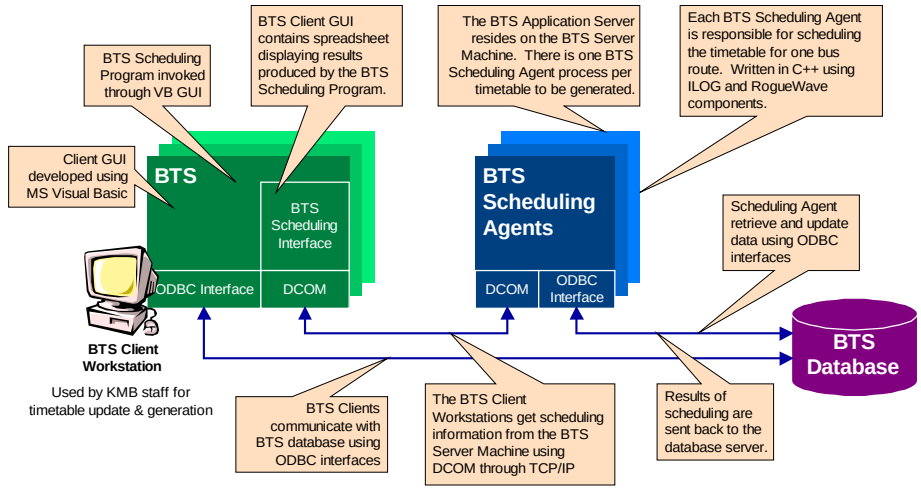

Fig. 3 Details of the BTS middleware infrastructure.

The client graphic user interface (GUI) is implemented using Microsoft Visual Basic. It communicates with BTS scheduling agents using DCOM and the BTS database using ODBC. Each request to generate a timetable from a client machine will fork a new scheduling agent process in the server machine. Each process will be responsible for the timetabling of one bus route. Both the client and application-server tier can communicate directly with the BTS database, which is implemented using the Microsoft SQL Server database.

5 The System Architecture

The following diagram illustrates the hardware architecture of the BTS. Only machines related to BTS are shown. The Bus Timetabling System operates within the Bus Company's corporate LAN. Timetables produced by the BTS are exported and transferred to corporate mainframe machines. The LAN connects all the remote BTS client workstations to the BTS server machine. BTS client workstations are installed at the corporate headquarters as well as several major bus depots. Each depot is responsible for producing timetables for its own bus routes.

The BTS server machine uses Microsoft NT Server as the operating system and executes a collection of BTS scheduling agent processes. This same machine contains the BTS SQL Server database.

The BTS client workstations use either Microsoft NT Workstation or Windows 98 as the operating system and execute the BTS Client Program. These machines can reside at any remote site and communicate with the BTS server machine through the Corporate LAN.

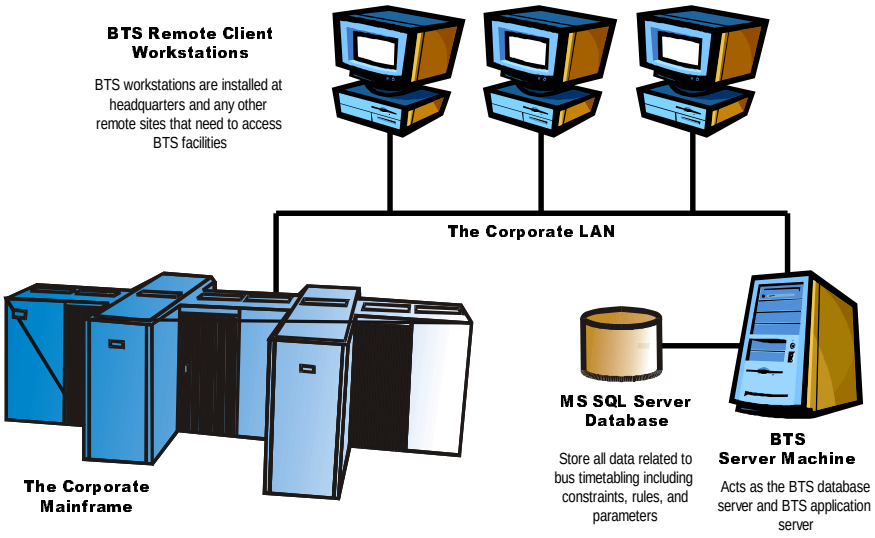

Fig. 4 The BTS system architecture.

6 Conclusion

This paper explained how we implemented distributed AI scheduling concepts using distributed object technologies to produce an advanced bus timetabling system. It outlined the methodologies and architectures used. Previously, the Bus Company used a semi-automatic program that requires many iterative tries before a satisfactory timetable can be produced. This is, of course, too time consuming and not efficient for their growing bus fleet. Our program generates a better schedule and in less time (within a minute). Previously, it took close to a day to process a single route.

Acknowledgement

The author would like to thank the Kowloon Motor Bus Company of Hong Kong for the cooperation received.

References

1. H.W. Chun, K.H. Pang, and N. Lam, “Container Vessel Berth Allocation with ILOG SOLVER,” The Second International ILOG SOLVER User Conference, Paris, July, 1996.
2. H.W. Chun, M.P. Ng, and N. Lam, “Rostering of Equipment Operators in a Container Yard,” The Second International ILOG SOLVER User Conference, Paris, July, 1996.
3. H.W. Chun, “A Distributed Constraint-Based Search Architecture For Bus Timetabling and Duty Assignment,” In the *Proceedings of the Joint 1997 Asia Pacific Software Engineering Conference and International Computer Science Conference*, Hong Kong, 1997.
4. E. Ephrati, “Divide and Conquer in Multi-agent Planning,” In *Proceedings of AAAI-94*, 1994.
5. ILOG, *The ILOG Solver 3.2 Reference Manual*, 1997.
6. K. Komaya and T. Fukuda, “A Knowledge-based Approach for Railway Scheduling,” *CAIA*, pp. 405-411, 1991.
7. V. Kumar, “Algorithms for Constraint Satisfaction Problems: A Survey,” In *AI Magazine*, 13(1), pp.32-44, 1992.
8. C. Le Pape, “Using Object-Oriented Constraint Programming Tools to Implement Flexible “Easy-to-use” Scheduling Systems,” In *Proceedings of the NSF Workshop on Intelligent*, Dynamic Scheduling for Manufacturing, Cocoa Beach, Florida, 1993.
9. H.-C. Lin and C.-C. Hsu, “An Interactive Train Scheduling Workbench Based On Artificial Intelligence,” In *Proceedings of the 6th International Conference on Tools with Artificial Intelligence*, November, 1994.
10. D.E. Neiman, D.W. Hildum, V.R. Lesser, and T.W. Sandholm, “Exploiting Meta-Level Information in a Distributed Scheduling System,” In *Proceedings of AAAI-94*, 1994.
11. J.-F. Puget, “A C++ Implementation of CLP,” In *ILOG Solver Collected Papers*, ILOG SA, France, 1994.
12. J.-F. Puget, “Object-Oriented Constraint Programming for Transportation Problems,” In *ILOG Solver Collected Papers*, ILOG SA, France, 1994.
13..G.L. Steele Jr., *The Definition and Implementation of a Computer Programming Language Based on Constraints*, Ph.D. Thesis, MIT, 1980.
14. P. Van Hentenryck, *Constraint Satisfaction in Logic Programming*, MIT Press, 1989.