# A DECISION SUPPORT SYSTEM (DSS) FOR THE RAILWAY SCHEDULING PROBLEM*

L. Ingolotti[1],P. Tormos[2] ,A. Lova [2],F. Barber [1],M.A. Salido [3] and M. Abril[1]
[1]*DSIC, Universidad Politecnica de Valencia,Spain;* [2]*DEIOAC, Universidad Politecnica de Valencia, Spain;* [3] *DCCIA, Universidad de Alicante, Spain*

**Abstract** The recent deregulation occurred in the public railway sector in many parts of the world has increased the awareness of this sector of the need for quality service that must be offered to its customers. In this paper, we present a software system for solving and plotting the Single-Track Railway Scheduling Problem efficiently and quickly. The problem is formulated as a Constraint Satisfaction Problem (CSP), which must be optimized. The solving process uses different stages to translate the problem into mathematical models, which are solved to optimality by means of mixed integer programming tools. The Decision Support System (DSS) we present allows the user to interactively specify the parameters of the problem, guarantees that constraints are satisfied and plots the optimized timetable obtained.

**Keywords:** Planning and Scheduling, Railway Scheduling Problems, Industrial Application of AI

## 1. INTRODUCTION

The recent deregulation in the public railway sector in many parts of the world has increased the awareness of this sector of the need for quality service that must be offered to its customers. Under pressure for improvement, computer tools have been developed to help planners do their work more efficiently and quickly. In this context, the timetable planning plays a fundamental role in the management and operation of a public transport system. Nowadays, software tools offer effective support for the construction of schedules. Many of the proposed tools are of the form of an interactive *what-if* application, in which the goal is to obtain feasible solutions quickly rather than obtaining an optimized solution. Information on network topology, engine properties as well as user requirements are stored in databases. Graphical user interfaces allow schedule planners to build and edit schedules interactively based on

*This work has been supported by a join contract RENFE-UC/UPV

time space diagrams that contain lines representing trains serving each route. However, the automatic generation of feasible schedules still remains too time-consuming. In particular, currently implemented algorithms are still too slow for networks of real-world size. In fact, existing software tools such as HASTUS, FBS, BERTA, MICROBUS, VISUM ÖV 7.0, ptv interplan, and solutions by TLC GmbH, Berlin, D are limited to only modifying an already existing timetable [4]. Although the use of such tools can be valuable, a train scheduling tool that is also capable of finding optimal solutions for the problem within a reasonable computational time is equally desired.

In this paper, a computer tool that is able to obtain optimized railway running maps[1] is presented. The running map will contain information regarding the topology of the network and the schedules of the trains; that is, arrival and departure times of trains at each station, frequency, stops, etc. The resulting optimized running map combines user requirements and deals with a wide variety of the complex constraints encountered in practical train scheduling.

## 2. THE TRAIN SCHEDULING PROBLEM

Planning of train schedules is a part of the general planning process of traffic systems and is traditionally broken down into several stages that have to be completed before a train schedule can be created. These stages are: Network Planning, Line Planning, Train Schedule Planning and Stock and Crew Planning. Planning rail traffic problems are basically optimization problems that are computationally difficult to solve as they belong to the NP-hard class of problems. Hence, efforts in the development of new, powerful, exact and heuristic algorithms are justified. The models and methods applied to solve these problems have been analyzed in [1], [3].

The majority of the papers published in the area of periodic timetabling in the last decade are based on the Periodic Event Scheduling Problem (PESP) introduced by [9]. Schrijver and Steenbeek in [8] developed a constraint programming based solver called CADANS to solve the feasibility problem. Nachtigall was the first to consider the objective function to be to minimize the passenger waiting time [5]. In [6] they developed a genetic algorithm to solve the problem in a context with two criteria taking into account investments in infrastructure over improvements in passenger waiting time. Odijk in [7] developed a cutting plane algorithm to solve the feasibility problem. His objective was to quickly generate a set of feasible timetables in order to be able to evaluate infrastructure projects. Recently, Caprara, Fischetti and Toth have proposed a graph formulation for the problem using a directed multigraph in which

[1]We consider a running map as an association between trains and the arrival and departure times at/from each location in their paths.

nodes correspond to departures/arrivals at a certain station at a given time instant [2]. This formulation is used to derive an integer linear programming model that is relaxed in a Lagrangian way. In their formulation, the objective is to maximize the sum of the benefits based on the differences between actual traversal times and ideal timetables, of the scheduled trains.

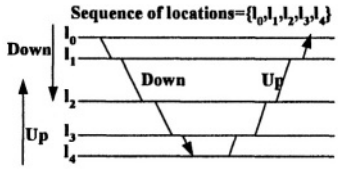


*Figure 1.* Each oblique line represents the position of a train depending on the time (axis x).

## 3. DESCRIPTION OF A SINGLE-TRACK RAILWAY SCHEDULING PROBLEM (STRSP)

Consider a STRSP problem defined by:

1 a set of ordered locations $L=\{l_0,l_1,...,l_m\}$ that must be visited by each train. Each $l_i$ is a place to stay or pass through. A pair of adjacent locations is joined by a single-way track.

2 a set of trains for each direction ($T_D$ and $T_U$). Given the sequence of locations L, $T_D=\{t_0,t_2,...,t_d\}$ visits the locations in the order given by the sequence (down direction), and $T_U=\{t_1,t_3,...,t_u\}$ visits the locations in the opposite order (up direction). The variable $t_i$ represents the ith train that starts the journey in a given direction (see Figure 1).

3 a journey for trains $T_U$ and $T_D$ in L specifies the traversal time for each section of track and each direction in L ($R_{i\rightarrow i+1}$ and $R_{i\rightarrow i-1}$), and the minimum stop time ($S_i$) for commercial purposes in each $l_i$.

Considering $t_y dep\_l_x$ and $t_y arriv\_l_x$ as the departure and arrival times of train $t_y$ from/at location $l_x$, the STRSP consists in finding the optimal running map (with minimum average traversal time) that satisfies all the following constraints[2]:

- Time Intervals to start the journey by the first train in each direction, $[min_D,max_D]$ and $[min_U,max_U]$,

  $min_D \leq t_0 dep\_l_0 \leq max_D \wedge min_U \leq t_1 dep\_l_m \leq max_U$

[2]The constraints are related to railway infrastructure, parameters of trains to be scheduled and traffic rules.

- Frequency Constraint specifies the period $(\mathbf{F_U/F_D})$ between departures of two consecutive trains in each direction at the same location,

$$\{\forall t_i, l_k / t_i \in T_D - \{t_d\} \wedge l_k \in L - \{l_m\}\}, t_{i+1} dep\_l_k = t_i dep\_l_k + F_D$$
$$\{\forall t_i, l_k / t_i \in T_U - \{t_u\} \wedge l_k \in L - \{l_0\}\}, t_{i+1} dep\_l_k = t_i dep\_l_k + F_U$$

- Stopover Constraint: a train must stay in a location $\mathbf{l_k}$ at least $\mathbf{S_k}$ time units,

$$\{\forall t_i, l_k / t_i \in T_D \cup T_U \wedge l_k \in L - \{l_m + l_0\}\}, t_i dep\_l_k \leq t_i arriv\_l_k + S_k$$

- Exclusiveness Constraint: a single-way section of track must be occupied by only one train at the same time,

$$\{\forall t_j, t_i, l_k / t_j \in T_D \wedge t_i \in T_U \wedge l_k \in L - \{l_m\}\},$$
$$t_i dep\_l_{k+1} \geq t_j arriv\_l_{k+1} \vee t_j dep\_l_k \geq t_i arriv\_l_k$$

A *conflict* occurs when two trains going in opposite directions require the same section of track at the same time. We denote $C_{ijk} \equiv < t_i, t_j, s_{k\_k+1} >$ when $t_i \in T_D$ and $t_j \in T_U$ compete for the section track $\mathbf{l_k}$,$\mathbf{l_{k+1}}$. The crossing of two trains going in opposite directions can be performed only at locations where one of the two trains has been detoured from the main track. This operation requires a reception and expedition time for the detoured train.

- Reception Time Constraint: At least are required $\mathbf{R_k}$ time units at location $\mathbf{l_k}$ between the arrival times of two trains going in the opposite direction (Figure 2),

$$\{\forall t_j, t_i, l_k / t_j \in T_D \wedge t_i \in T_U \wedge l_k \in L\},$$
$$t_j arriv\_l_k \geq t_i arriv\_l_k + R_k \vee t_i arriv\_l_k \geq t_j arriv\_l_k + R_k$$

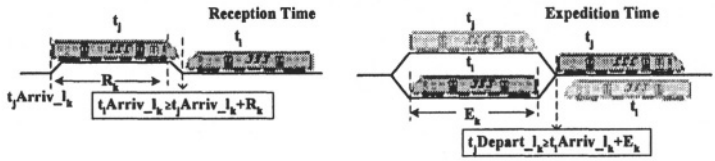


*Figure 2.* Reception and Expedition time constraint

- Expedition Time Constraint: At least are required $\mathbf{E_k}$ time units at location $\mathbf{l_k}$ between the arrival and departure times of two trains going in the opposite direction (Figure 2),

$$\{\forall t_j, t_i, l_k / t_j \in T_D \wedge t_i \in T_U \wedge l_k \in L\},$$
$$t_j dep\_l_k \geq t_i arriv\_l_k + E_k \vee t_i dep\_l_k \geq t_j arriv\_l_k + E_k$$

- Precedence Constraint: each train employs a given time interval ($R_{k\to k+1}$ *or* $R_{k\to k-1}$) to traverse each section of track ($l_k \to l_{k+1} or l_k \to l_{k-1}$) in each direction,

$$\{\forall t_i, l_k / t_i \in T_D \wedge l_k \in L - \{l_m\}\}, t_i arriv\_l_{k+1} = t_i dep\_l_k + R_{k\to k+1}$$
$$\{\forall t_i, l_k / t_i \in T_U \wedge l_k \in L - \{l_0\}\}, t_i arriv\_l_{k-1} = t_i dep\_l_k + R_{k\to k-1}$$

The complexity of the problem lies mainly in the number of conflicts that can appear during the generation of the running map. In each conflict, one of two trains must wait for the release of the section of track (priority assignment). This problem is a well known NP-hard problem which makes exploring all possibilities for optimality complex and inefficient. In the DSS system, the search space is drastically reduced by means of a pre-processing stage before applying CSP techniques to solve the problem.

# 4. THE SOLVING TOOL: A DEPENDENT-DOMAIN CSP

The architecture of our system is shown in Figure 3. First, the user gives the parameters of a required running map (L1): which journeys, number of trains, time interval to start the journey and frequency. All parameters are stored in a common database and used by the solver process (L2). Finally, the solution is shown graphically to the user (L3), who can interact with the system.

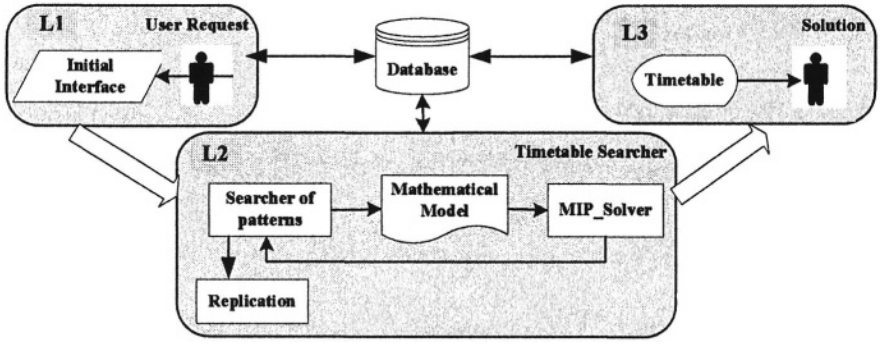


*Figure 3.* General System Architecture

The main module is the solver process, which obtains an optimized running map (L2 in Figure 3). This process is efficiently performed by identifying, solving, and replicating a given pattern.

## 4.1 Identification of replication patterns

The identification of replication patterns to reduce complexity in order to solve STRSP is based on the concept that we have named *Equivalent Conflicts:*

Two conflicts $C_{mek}$ (conflict between $t_m$ and $t_e$ for the section of track $k, k+1$) and $C_{ojh}$ (conflict between $t_o$ and $t_j$ for the section of track $h, h+1$) are equivalents if and only if:

- they occur in the same section of track (i.e.: $k = h$);
- for each conflict $C_{mm'v}$ (i.e.: each conflict of $t_m$ with another train $t_{m'}$ for a section track $v, v+1$) there exists one conflict $C_{oo'v}$ (conflict between $t_o$ and another train $t_{o'}$ for a section track $v, v+1$) or vice versa;
- for each conflict $C_{e'ew}$, a conflict $C_{j'jw}$ exists or vice versa.

For instance, in Figure 4, $C_1$ and $C_2$ (which represent to $C_{mek}$ and $C_{ojh}$ respectively) are equivalent conflicts due to the following:

- They occur in the same section of track.
- all conflicts between $t_m$ and any other $t_{m'}$ ($C_3$, $C_5$) occur in the same section of track that all conflicts between $t_o$ and $t_{o'}$ ($C_4$, $C_6$), respectively.
- all conflicts between $t_e$ and any other $t_{e'}$ ($C_7$) occur in the same section of track as all conflicts between $t_j$ and $t_{j'}$ ($C_8$), respectively.

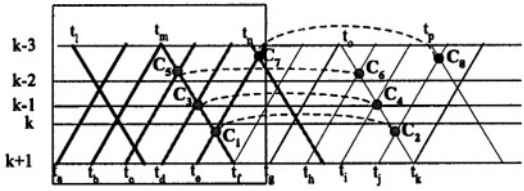


*Figure 4.* An example for equivalent conflicts and pattern identification

The concept of equivalent conflicts allows us to identify patterns in a running-map. A pattern is a part of the whole running map, where only non equivalent conflicts exist (Figure 4). Each possible start departure time in each direction gives rise to a set of non-equivalent conflicts and a consequent pattern. Solving a pattern implies solving the complete running map, because if the related set of non-equivalents conflicts is solved, all equivalent conflicts may be solved similarly. That is to say, the complete running map can be obtained by replicating the solved pattern. Finding and replicating the pattern with minimum cost produces the optimal running map. Thus, the basic process consists in identifying and solving the patterns for each possible start departure time in each direction (Figure 5 shows how a pattern for a given start departure time is identified and solved), and choosing the pattern with minimum cost and replicating

it along the running map. It is important to remark that once a pattern is identified, increasing the number of trains does not increase the problem complexity.

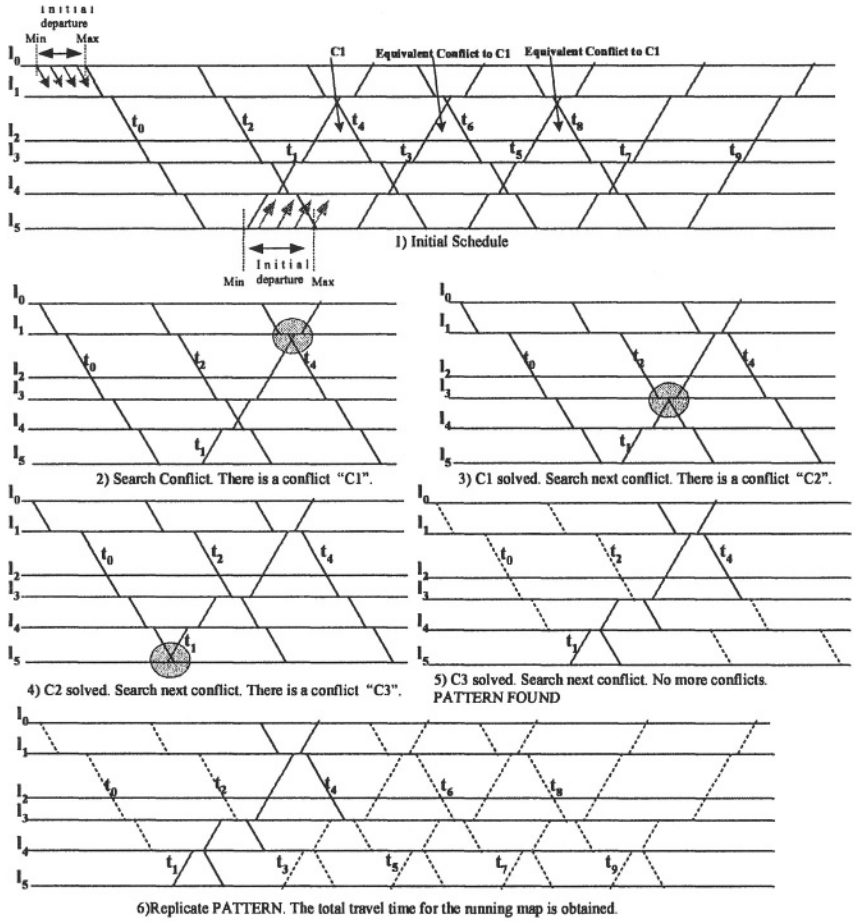


*Figure 5.* Outline of the algorithm

## 4.2 Algorithm for Single-Track Railway Scheduling

The pattern to be solved and replicated depends on the set of non-equivalent conflicts which in turn depends on the initial departure of the trains. The algorithm to identify and to solve patterns has three nested loops (Figure 6): *Loop 1* explores departure times for the first trains in each direction, *Loop 2* explores assignment of priorities for trains and *Loop 3* identifies the set of non-equivalent conflicts. Constraints are generated only for the subset S of trains

implied in the current set of non-equivalent conflicts, and solved (if possible) according to the assignment of priorities given by the second loop.

Once all possible sets of non-equivalent conflicts (patterns) are identified and solved, the best solution is replicated to obtain the optimal running map.

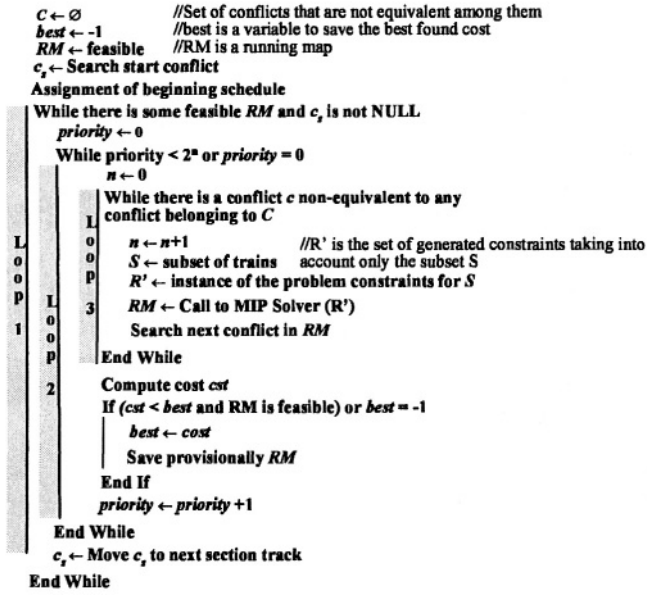


*Figure 6.* Main steps in the algorithm

## 5. CAPABILITIES OF THE DSS FOR THE STRSP

The DSS developed in this work is a tool for solving a STRSP according to the values specified by the user for given parameters: frequency, number of trains, journey, and start time interval, for each direction. The system considers the set of parameters with their corresponding values as a request. Once the problem has been parameterized, the system solves it, saving the obtained running map in a database for its later graphical display. Figure 7 shows an example of a request solved by the DSS. In this example, for each direction the user composes his request choosing: a frequency (1 hour), a journey, a number of trains (100) and a time interval to start the journey (6:00-6:30 for DOWN and 8:00-8:30 for UP). The system took 25 seconds to solve this problem using an Intel Pentium 4 1,6 GHz processor. The user interface of the DSS allows the parameters to be easily modified obtaining a solution for each request. This makes the DSS an appropriate tool application for what-if analysis. The number of trains that could be allocated in one day in the running map with a given frequency, or the best departure time to start a given journey, in order to obtain

the minimum average traversal time, etc, are some examples of useful information for the final operator. The graphical interface through the space-time diagrams shows the topology of the solution in a clearer way.

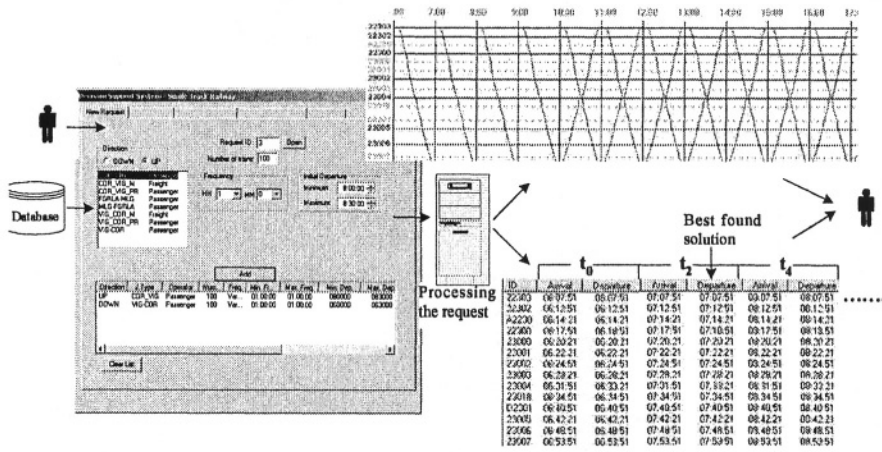


*Figure 7.* An example using the application

## 6. RESULTS

In this section we evaluate some problems, whose value parameters can be applied to practical cases. Each constraint satisfaction problem is defined by the pair $< n, f >$, where $n$ is the number of trains and $f$ is the frequency between consecutive trains. In Table 1, we present the execution time in problems solved with DSS and general constraint solvers (LINGO, CHEEP, CPLEX, etc), where the number of trains was increased from 10 to 50 while the frequency was fixed to 60 and problems where the frequency was increased from 60 to 120 and the number of trains was fixed to 20. In both cases, the number of locations was 40 and the departure time was [06:00:00 - 06:30:00] in down direction and [05:30:00 - 06:30:00] in up direction. The problems were solved with an Intel Pentium 4 1,6 GHz processor.

In Table 1, we observe that DSS reduces considerably the execution time in all cases due to it reduces the search space using the heuristic described in previous sections while general solvers provide the optimal schedule without preprocessing and they must study the complete problem.

## 7. CONCLUSIONS AND FUTURE RESEARCH

In this paper, a DSS to solve the Single Track case of this problem has been developed. The system provides the user with an interface to parameterize the

*Table 1.* Execution time in problems $< n, 60 >$ and $< 20, f >$.

| Problems | DSS - Run Time (sc) | Solvers - Run Time (sc) | Problems | DSS - Run Time (sc) | Solvers Run Time (sc) |
|---|---|---|---|---|---|
| <10,60> | 3 | 9 | <20,60> | 5 | 313 |
| <16,60> | 4 | 46 | <20,75> | 1 | 4200 |
| <20,60> | 5 | 313 | <20,90> | 1 | 95 |
| <36,60> | 6 | 1487 | <20,105> | 1 | 34 |
| <50,60> | 7 | >4800 | <20,120> | 1 | 147 |

problem. It solves the request in an optimal and efficient way using CSP guided by the proper characteristics of the problem, thereby reducing the search space and hence the computational effort. The user can obtain different running maps for easier and more effective decision making. The current system is being validated by a railway company. Future work will address to the extension of the method in order to deal with double tracks and networks already occupied by other trains.

## References


[1] Bussieck, M.R., Winter, T., and Zimmermann, U.T., 'Discrete optimization in public rail transport', *Math. Programming,* **79(1-3)**, 415–444, (1997).

[2] Caprara, A., Fischetti, M., and Toth P., 'Modeling and solving the train timetabling problem', *Operations Research,* **50**, 851–861, (2002).

[3] Cordeau, J.F., Toth, P., and Vigo, D., 'A survey of optimization models for train routing and scheduling', *Transportation Science,* **32(4)**, 380–404, (1998).

[4] Liebchen, C., and Möhring, R., 'A case study in periodic timetabling', *Electronic Notes in Theoretical Computer Science, 66,* **6**, (2002).

[5] Nachtigall, K., 'Periodic network optimization with different arc *frequencies' Discrete Applied Mathematics 69,* **1-2**, 1–17, (1996).

[6] Nachtigall, K., and Voget, S., 'Minimizing waiting times in integrated fixed interval timetables by upgrading railway tracks', *European Journal of Operational Research,* (1997).

[7] Odijk, M.A., 'A constraint generation algorithm for the construction of periodic railway timetables', *Transportation Research, 30,* **6**, 455–464, (1996).

[8] Schrijver, A., and Steenbeek, A., 'Timetable construction for railned', Technical Report, *CWI, Amsterdam, The Netherlands (in Dutch),* (1994).

[9] Serafini, P., and Ukovich, W., 'A mathematical model for periodic scheduling problems', *SIAM Journal on Discrete Mathematics,* **2(4)**, 550–581, (1989).