Control and Cybernetics

vol. **22** (1993) No. 4

Parallel computations for decision support systems

by

Ignacy Kaliszewski

Systems Research Institute
Polish Academy of Sciences
ul. Newelska 6, 01-447 Warszawa
Poland

The problem of effective implementations of computing intensive Decision Support Systems is discussed. Background computations and parallel computations are proposed as means to overcome the computing capacity bottleneck in computer supported interactive decision making. The transputer technology is envisaged as appropriate for multiprocessor implementations of Decision Support Systems.

Keywords: decision support, background computations, parallel computations, Pareto analysis, multiobjective optimization, multiprocessor networks.

1. Introduction

Most recent *Decision Support Systems* (DSSs) , both general purpose or specialized, have two futures in common: first - they are *interactive*, second - they are *computer based* (cf eg Lewandowski, Wierzbicki 1989, Dror et al. 1991). An interactive decision support system is a system designed and implemented to assist interactive decision making. An interactive decision making is a decision process split into several stages during which *Decision Maker* (DM) progressively expresses his preferences, analyses trial decisions and learns about the structure of a decision problem. One interaction (iteration) of such a process consists of the *decisi n phase* (DM is active, the computer is idle in the sense that it performs no bulky computations but only uses its output devices to present appropriate information in textual and/or graphic form) and the *computing phase* (DM is inactive, the computer is active). Although efficient implementations of DSSs have become possible by intensive use of computers, computers alone can be bottlenecks as the size and complexity of decision problems grows. Obviously, the duration of any computer phase in interactive decision making must be kept within reasonable limits. It happens, however, that even for medium size decision problems the time consumed for determining one trial decision is significant and in such situations the practical usefulness of DSSs might be questionable. Below we propose two remedies for this.

2. Background computations

The first remedy we propose is the idea of *background computations.* It is a generic feature of all interactive DSSs that during the decision phase for most of the time the computer is idle. Even if DM activates the computer to store, sort, retrieve, compare previously derived decisions, and display the related information, this usually consumes a negligible part of computer capacity. The surplus capacity can be used to activate possible extensions or options of a DSS, which to keep the duration of individual computing phases within reasonable limits have been kept inactive. If there is no extra charge for using the surplus capacity or the charge is low (which is true eg for personal and dedicated computers) additional processing can be started even if DM may later show no interest in the results. Usually, it takes some time for DM to make a judgment about the current trial decision and to set guidelines for the search for the next trial decision. During this time, which is otherwise lost, additional processing can be significantly advanced if not completed.

It is important that all the background computations should not harass DM in his process of decision making (eg information presented on the screen should not be affected). The idea of background computations is well known in *multitasking computer systems*, where several *tasks (processes)* started by one or several users can be processed *concurrently*. Concurrency means that tasks are processed one by time but interchangeably, where the processor after some time spent on processing a task suspends it and starts (or resumes) processing a subsequent task. Usually tasks are structured by some priority rules. It is possible then to interact with one task (which is said to be in the foreground) whereas the remaining tasks run in the background. Tasks are to be managed on the computer system level and this is hardly achievable by an ordinary user. It is therefore necessary to organize background computations from the level of a program. This calls for a capability of software to create submodules of a program, called threads, which can be processed concurrently. Mechanisms of this type are present in several algorithmic languages as Ada, Modula, and various "parallel" extensions of C, Fortran, and Pascal.

3. Parallel Computations

If it happens that the computer capacity does not match decision support system requirements, then the next possible step is to switch to *parallel computations.* Parallel computations have become a reality with the emergence of multiprocessor computers. In computers of this sort threads can be physically distributed among several processors. This may result in a speed-up of computations with the theoretical bound on the speed-up equal to the number of processors used.

Though some academic and even commercial multiprocessor computers are now available, a limited access to such installations and/or a high cost of their services make them hardly advisable in DSSs. One must remember that most implementations of DSSs have been done with desk-top minicomputers.

4. Multiprocessing on Transputer Networks

Quite recently a technology has emerged which is well suited to the needs of decision making and solves the problem of ensuring sufficient computer capacity for successful implementations of DSSs. It features a family of microprocessors, called *transputers* (Relofs 1987, Whitby–Stevens, Hodgkins, 1990), each with four links, which can be connected via links with other transputers into a network. Moreover, the whole network can be connected via an idle link to any computer turning it into a multiprocessor computer of significant capacity. A transputer (T800 version) is a 32-bit chip operating with 30MHz internal clock. It has 4 Kb on-chip memory and has an address space for an external memory up to 4 Gb. A key to the success of transputers is the speed of transmission via links: links are autonomous to transputer's CPU (can operate concurrently with the CPU) which results in low communication overheads even if all four links are running at the same time. The effective speed of unidirectional transmission is 1.8 Mb/sec. There is no limit on the number of transputers working in a network. Transputer networks can be programmed with parallel extensions of C, Fortran, Pascal (cf eg Parallel C, User Guide, 1991) or assembler-like Occam. A preliminary application of PC based transputer networks to solve multiobjective optimization problems (a formal model for many decision problems) has been already reported (Kaliszewski, 1990).

5. An Example - A Pilot DSS Implementing Quantitative Pareto Analysis on a Network of Transputers

Quantitative Pareto Analysis (Kaliszewski 1993) is a coherent methodology providing DM with a variety of information about admissible decisions whenever the multiobjective optimization problem is the underlying formal model for decision making. Quantitative Pareto Analysis provides a methodological framework for:

- generation of efficient decisions,

- derivation of numerical information on efficient decisions such as values of criteria, distances to a certain ideal (may be fictitious) decision, maximal and minimal values of separate criteria over the admissible decision set,

- derivation of a certain natural, hierarchical structure over the set of efficient decisions,

- visualization of decision making processes by offering a method for fast approximation of sets of efficient outcomes (Pareto sets),

- values of trade-offs,

- sensitivity analysis of efficient decisions with respect to perturbations of utility functions,

- sensitivity analysis of efficient decisions with respect to perturbations of objective functions.

The first two items are standard elements implemented explicitly or implicitly in any DSS. All the remaining items of Quantitative Pareto Analysis result from interpretations of specific numerical characterizations of efficient decisions related to the notions of proper efficiency and substantial efficiency (Kaliszewski 1993). The analysis is updated each time a new trial decision is generated in the course of interactive decision making. Quantitative Pareto Analysis can be applied in its full extend in an interactive decision making method to enhance the quality of decisions. At any stage of an interactive decision making process DM is free in selecting from the whole variety of information provided by Quantitative Pareto Analysis the information he needs. The analysis (especially calculations of values of trade-offs) is rather demanding in computing capacity and computation time. Therefore, provided the computer used to build a DSS is parallel, all the respective items of the analysis are to be realized as soon as a trial decision has been generated, even if results of the analysis for this particular decision will not be later used by DM. A pilot DSS implementing Quantitative Pareto Analysis is currently tested in the Mathematical Programming Department of the Systems Research Institute. At present only linear multiple criteria decision making problems can be approached by this methodology. A hardware platform for the system is a network of four transputers hosted by a PC computer.

Below we shall present in detail implementation issues. To this aim we have to present the underlying problem, ie Quantitative Pareto Analysis more closely. However, to describe Quantitative Pareto Analysis in its full extent we would need another paper, so here we confine ourselves to one element of it, namely to calculations of trade-offs.

5.1. Calculation of trade-offs

The notion of *trade-offs* plays a key role in multiobjective optimization, Many interactive decision making algorithms use this notion to steer the process of decision making.

The definition of a multiobjective programming problem uses the notion of *decision space* X , *feasible set* $X_0 \subseteq X$, *outcome space* Y , *objective functions,* and *outcome set* $Z = f(X_0) \subseteq Y$. To define trade-offs it is enough to confine oneself to the notion of outcome set. We make no specific assumption about outcome sets except that they are compact.

Let $\bar{y} \in Z$, $Z \subseteq \mathcal{R}^k$, Z is an outcome set. Throughout this paper we assume that Z is compact, ie closed and bounded. For all $i = 1, ..., k$, denote

$$Z_i^{<}(\bar{y}) = \{y \in Z \mid y_i < \bar{y}_i \, , \; y_l \geq \bar{y}_l \, , \; l = 1, ..., k \, , \; l \neq i\} \, ,$$

$$Z_i^{\leqslant}(\bar{y}) = \{y \in Z \mid y_i \leq \bar{y}_i \, , \; y_l \geq \bar{y}_l \, , \; l = 1, ..., k \, , \; l \neq i\} \, .$$

DEFINITION 5.1 *Let $\bar{y} \in Z$. The* **global trade-off** $T_{ij}^G(\bar{y})$ *involving i and j, $i,j = 1,...,k$, $i \neq j$, is*

$$\sup_{y \in Z_j^<(\bar{y})} \frac{y_i - \bar{y}_i}{\bar{y}_j - y_j}.$$

THEOREM 5.1 *Given an element $\bar{y} \in Z$, the trade-off $T_{ji}^G(\bar{y})$ exists if and only if $Z_i^<(\bar{y}) \neq \{\emptyset\}$, and there exist a number ρ, $\rho > 0$, such that $\bar{y}$ solves the problem*

$$\max_{y \in Z_i^{\leqslant}(\bar{y})} ((1+\rho)y_i + \rho y_j). \tag{1}$$

If such a ρ exists, then

$$T_{ji}^G(\bar{y}) \leq (1+\rho)\rho^{-1}$$

REMARK 5.1 *Observe that if $Z_i^<(\bar{y}) = \{\emptyset\}$, then, by the definition, the trade-off $T_{ji}^G(\bar{y})$ does not exist but $\bar{y} \in Z_i^{\leqslant}(\bar{y})$ may solve the problem (1) as it is eg when $Z_i^{\leqslant}(\bar{y}) = \{\bar{y}\}$.*

THEOREM 5.2 *Given an element $\bar{y} \in Z$ for which the trade-off $T_{ji}^G(\bar{y})$ exists. Then either there exists the maximal value $\bar{\rho}$ of the parameter ρ, $\rho > 0$, for which $\bar{y}$ solves the problem (1), ie the problem*

$$\max_{y \in Z_i^{\leqslant}(\bar{y})} ((1+\rho)y_i + \rho y_j)$$

and

$$T_{ji}^G(\bar{y}) = (1+\bar{\rho})\bar{\rho}^{-1}$$

or $\bar{y}$ solves the above problem for every nonnegative ρ and

$$T_{ji}^G(\bar{y}) \leq 1.$$

THEOREM 5.3 *Given an element $\bar{y} \in Z$ for which the trade-off $T_{ji}^G(\bar{y})$ exists. Suppose that $\bar{y}$ solves the problem (1), ie the problem*

$$\max_{y \in Z_i^{\leqslant}(\bar{y})} ((1+\rho)y_i + \rho y_j)$$

for every positive ρ. Then either there exists the minimal value $\tilde{\rho}$ of the parameter ρ, $\rho \geq 0$, for which $\bar{y}$ solves the problem

$$\max_{y \in Z_i^{\leqslant}(\bar{y})} (\rho y_i + (1+\rho)y_j) \tag{2}$$

and

$$T_{ji}^G(\bar{y}) = \tilde{\rho}(1+\tilde{\rho})^{-1}$$

or $\bar{y}$ does not solve the above problem for every nonnegative ρ and

$$T_{ji}^G(\bar{y}) = 1.$$

Values of trade-offs constitute an important information but DM is free to use it or not. In particular, he may wish to know values for particular pairs of indices.

Since, depending on the size of the problem, calculations of trade-offs may be time consuming (in our case one LP problem and , in the worst case, two parametric problems have to solved for one pair of indices) we have decided to calculate all $n(n-1)/2$ trade-offs in the background and, since our instalation permits that, in parallel.

5.2. Granuality of Parallel Computations

An important issue we had to decide about was the granuality of parallel computations in the system. Though there is no precise definition, granuality is regarded as *fine* if parallelism is introduced at the level of operations and *coarse* if it is introduced at the level of procedures.

The basic building block of numerical computations in the system is a LP solver. It is intensively used in calculations of trade-offs (see Section 5.1.) and in the other elements of Quantitative Pareto Analysis. It is evident then that the issue of parallelism should be considered for the solver first.

The LP solver we use is an implementation of the simplex method. The simplex method cannot be efficiently parallelized since it contains sequences of operations which are inheritely sequential (Wallach 1982). Moreover, fine granuality is efficently implemented on installations built of either specialized processing units (vector processors, arthmetic coprocessors) or many simple homogenous processors (as in systolic arrays). Hence, we have decided in favour for coarse granuality and we have not attempted to parallelize the LP solver itself. This means that linear programming problems are treated in the system as separate (sequential) tasks which in turn are computed in parallel.

5.3. Computer Farm versus Multi Task Applications

Another important issue is the model of parallel computations. High level algorithmic languages for transputer programmning such as Parallel Fortran, Parallel Pascal, Parallel C (the system is implemented in Parallel C) support two models: *computer farm* and *multiple task application*. In computer farm there are two types of tasks: a *master* and *slaves*. The master initializes computations, sends and receives data from slaves which perform the bulk of computations. Slaves share the same set of instructions and their action is differentiated by input data. An important future of that model is that tasks are assigned to the physical processing units, ie transputers, automatically by a special program called *router*. The logic of assigning is hidden to the programmer and the router works on installations with any number of transputers, starting from one-transputer installations.

Much more control is given to the programer in the second model where the programmer is responsible for the physical assignment of tasks to transputers and tasks need not to be homogenous as in the case of slaves.

The implementation of the decision support system is rather complex, thus it would be difficult to put all it functions in the framework of one homogenous task (slave) as it is required in the computer farm model. Therefore, we have selected the more laborious but more flexible multi task application model.

5.4. Windows enviroment

The system produces a range of numerical and graphical information which should be conveniently presented to DM. There is a strong need to work in Windows enviroment to control the access to the data and results. The idea behind Quantitative Pareto Analysis is that DM should be presented with a spectrum of information from which he can select at wish, so opening, closing and updating windows is of primary importance for data and results effective handling.

To provide an interface between transputers and Windows enviroment the system makes use of Nexis Window File Server.

5.5. Software Specification

The following software has been used to develop the system:

- Parallel C (a parallel extension of C) by 3L;
- tGRAPH (transputer graphic) by Rahmonic Resources Pte Ltd;
- Nexis Windows File Server (interface for transputers and Windows by Nexis;
- Windows by Microsoft.

6. Conclusions

The system is a powerful tool for deriving quantitative data in decision making processes which for the underlying model have multiobjective programming problems. Various decision making methods can be easily implemented upon this system.

Quantitative Pareto Analysis is a general framework for quantitative analysis in multiobjective programming. For LP models there exist specialized methods for such issues as eg sensitivity analysis which obviously outperform the general approach. The full strength of Quantitative Pareto Analysis can be observed when it is applied to models of multiobjective programming other than linear. We intend to extend the system for other models and linear integer problems will be our first step in this direction.

References

DROR M., SHOVAL P., YELLIN A. (1991) Multiobjective linear programming: another DSS. Decision Support Systems, 7, 221–232.

KALISZEWSKI I. (1990) Determination of maximal elements in finite sets on a network of transputers. Systems Research Institute Technical Report ZPM2/90, Warszawa.

KALISZEWSKI I. (1992) Quantitative Pareto Analysis by Cone Separation Technique. Systems Research Institute Technical Report ZPM19/92.

KALISZEWSKI I. (1993) Quantitative Pareto Analysis and the principle of background computations. In: User Oriented Methodology and Techniques of Decision Support, J. Wessels, A.P. Wierzbicki (eds), Lecture Notes in Economics and Mathematical Systems, 397, 112–120, Springer Verlag, Berlin.

LEWANDOWSKI A., WIERZBICKI A.P. (EDS), (1989) Aspiration Based Decision Support Systems, Theory, Software, Applications. Lecture Notes in Economics and Mathematical Systems, 331, Springer Verlag. Berlin.

PARALLEL C, USER GUIDE, (1991) 3L Ltd.

ROELOFS B. (1987) The transputer. A microprocessor designed for parallel processing. *Micro Cornucopia*, **38**, 6–8.

WALLACH Y. (1982) Alternating Sequential/Parallel Processing. Springer Verlag, Berlin, Heidelberg, New York.

WHITBY–STEVENS C., HODGKINS O. (1990) Transputers - past, present and future. *IEEE Micro*, 19-19, 76-82.

Control and Cybernetics
Volume **21**, 1992

Alphabetical list of contributors

J. BABAROWSKI, Z. NAHORSKI AND A. STRASZAK, Oil Prices and Unemployment Relationships, 2, 149–168.

Y. ALY MOHAMED BAGHDADI, On the convergence of a conditional ϵ–subgradient method for nondifferentiable convex optimization, 3/4, 67–77.

S.J. BRAMS AND S. MERRILL, III, Arbitration procedure with the possibility of compromise, 1, 131–149.

BUI CONG BUONG, Vietnam, A Hybrid Transitional Model, 2, 85–103.

T.A. DARLING, J.L. MUMPOWER, Simulating process and outcome for two–party contract negotiations, 1, 151–164.

P. FLEISSNER AND G. KRÜGER, Simulated Economic and Social Futures of the Late German Democratic Republic by Means of a System Dynamics Model, 2, 67–83.

J. GADOMSKI, Production in a Non–Market Economy, 2, 105–130.

J. GADOMSKI, C. IWAŃSKI AND P. WYCZAŃSKI, Forecasting the Polish Zloty/US Dollar Free Market Exchange, 2, 169–178.

J. GAWIN, P. FOCHTMAN, Substitution algorithm for binary chains, 3/4, 93–100.

W. GÜTH AND P. OCKENFELS, Game theoretic analysis of bargaining models, 1, 185–229.

N. HOWARD, Environmental negotiations: the problem of time–inconsistent strategies, 1, 61–83.

K. IVANIČKA A Simulation Model of Development of a Group of Companies, 2, 179–192.

L. JÓDAR, E. PONSODA, Analytic approximate solutions and error bounds for linear matrix differential equations appearing in control, 3/4, 21–34.

D.M. KILGOUR, K.W. HIPEL and L. FANG, The graph model for conflicts as a negotiation support tool, 1, 85–103.

Z. KOPERCZAK, S. MATWIN AND S. SZPAKOWICZ, Modelling negotiation strategies with two interacting expert systems, 1, 105–130.

S. LAHIRI, Threat bargaining problems with incomplete information, 1, 267–275.

J.–P. LEONARD, The Possibility of a Soft Revolution, 2, 31–49.

W.F.G. MASTENBROEK, Negotiating: efforts towards integrative concepts, 1, 5–19.

J. MERCIK, Voting Procedure Characteristics, 2, 207–215.

P. MOORE, Restructuring State–Owned Major Industries: A Comparative View, 2, 131–147.

A. OKADA, Noncooperative bargaining anf the core of an n–person characteristic function game, 1, 231–250.

J. W. OWSIŃSKI, A Metaphoric System of Economy, 2, 23–29.

J. W. OWSIŃSKI AND S. ZADROŻNY, Micropolitical Behaviour: the Case of Polish Parliamentary Elections of 1991, 2, 217–236.

T. RAJASALU, The Estonian Economy and Its Development Problems, 2, 51–65.

A. SAMBANDAN, S.L. HAKKAPAKKI, Optimal pole assignment and redundancy problem, 3/4, 49–65.

F. SEO AND I. NISHIZAKI, On construction of a cooperative fuzzy game in international fuzzy environment: a possibilistic approach, 1, 277–294.

I. SPECTOR, Developing the negotiator's tool box: practical systems to support effective international negotiations, 1, 21–33.

J. STEFAŃSKI, Bargaining in changing environments,1, 251–265.

A. STRASZAK The Background for Transition: the Case of Poland, 2, 13–21.

F. TURNOVEC, Power Distribution in Czecho–Slovak Parliaments, 2, 193–205.

R. J. VILLANUEVA, L. JÓDAR, Stable closed form solution of a class of Riccati matrix differential equations, 3/4, 7–20.

M. WANG AND K.W. HIPEL, Misperceptions and bargaining in the Persian Gulf crisis, 1, 35–60.

A. YASTREBOV, M. GRZYWACZEWSKI, Design of multistep algorithms and local optimal input for dynamic system identification, 3/4, 35–47.

A. ŻOŁNIEREK, The experimental investigations of pattern recognition algorithms for second–order Markov chains, 3/4, 79–91.

Control and Cybernetics
Volume **22**, 1993

Alphabetical list of contributors

R. ANTKIEWICZ, T. NOWICKI, A computer memory sharing problem, 1/2, 71–77.

C.B. BASNET, D.N. CLARK, L.R. FOULDS, A decision support system approach to milk tanker routing, 4, 131–144.

$\boxed{\text{A. BLAQUIÈRE,}}$ C-optimal threat decision n-tuples in collective bargaining, 4, 63–75.

M. CHUDY, Cost characteristics of system service with input controlled by Markov chain, 1/2, 55–61.

Z. CZERWIŃSKI, W. JUREK, E. PANEK, A medium–term model of Polish economy. An approach to transition problems, 1/2, 9–28.

S. DENIZIAK, K. SAPIECHA, A system for behavioral level synthesis of PLD–based circuits, 3, 105–113.

M. FEDRIZZI, J. KACPRZYK, H NURMI, Consensus degrees under fuzzy majorities and fuzzy preferences using OWA operators, 4, 77–86.

S. GUSZKOWSKA, Description of a subset of Polish natural language: a mathematical model, 1/2, 85–105.

Y.Y. HAIMES, Risk of extreme events and the fallacy of the expected value, 4, 7–31.

P. HOLNICKI, A. KAŁUSZKO, A. ŻOCHOWSKI, A multilayer computer model for air quality forecasting in urban/regional scale, 3, 5–28.

W. HOMENDA, An experiment in computer control of the learning process, 1/2, 113–121.

I. KALISZEWSKI Parallel computations for decision support systems, 4, 145–152.

E. KĄCKI Some remarks on the school of Edmund Lipiński (1922–1990), 1/2, 81–84.

L. KLUKOWSKI, Time series forecasting on the basis of Markovian piecewise linear trend models, 1/2, 29–44.

L. KRUŚ, P. BRONISZ, B. ŁOPUCH, MCBARG system supporting multicriteria bargaining, 4, 119–128.

R. KULIKOWSKI, Models of Motivation, Satisfaction and Successful Planning in R & D, 4, 33–53.

D.B. MEISTER, N.M. FRASER, Towards a prescriptive negotiation support system, 4, 103–117.

T. NOWICKI, Optimal periodic buffer exhaustion strategy, 1/2, 63–69.

Y. QIU, J. WU, Non–linear intertemporal dilemma models, 4, 55–61.

J. RATYŃSKA, Mathematical model of natural language phrase generation, 1/2, 107–111.

G. RAY, L. SUBRAHMANYAM, Modelling hierarchical dynamic systems for decentralized state estimation, 3, 29–45.

NGUYEN N. SAN, The system — assumed–model — reduced–model relationship: the optimal projection equations with input error, 3, 47–68.

S. STORØY, A method for improving weights in column aggregated linear programs, 1/2, 45–53.

S. TRYBUŁA, A noisy duel with two kinds of weapons, 3, 69–103.

R. VETSCHERA, Feedback–oriented group decision support, 4, 87–102.

J. WOJUTYŃSKI, Automatic analysis of spatial electrical circuits with graph theory, 1/2, 123–130.