STUDIA INFORMATICA 2010
Volume 31 Number 2B (90)

Mateusz WIBIG
Deloitte Business Consulting, Technology Integration

OPTIMISATION OF BUSINESS PROCESSES USING PETRI NETS AND DYNAMIC PROGRAMMING

Summary: This paper describes the idea of improving the simulation-based optimization of business processes using the dynamic programming. The Petri Nets scalability together with the concept of dynamic programming is used in attempt to reduce the number of necessary computations while applying the changes into the process.

Keywords: genetic algorithm, Petri nets, expert system, dynamic programming, business processes

OPTYMALIZACJA PROCESÓW BIZNESOWYCH PRZY POMOCY SIECI PETRIEGO I PROGRAMOWANIA DYNAMICZNEGO

Streszczenie. Artykuł przedstawia ideę usprawnienia, opartą na symulacji metody optymalizacji procesów biznesowych, przy pomocy programowania dynamicznego. Skalowalność sieci Petriego i pochodząca z programowania dynamicznego koncepcja dzielenia problemu na mniejsze podproblemy wykorzystana została w próbie zmniejszenia liczby operacji niezbędnych do wdrożenia zmian w procesie.

Słowa kluczowe: algorytm genetyczny, sieci Petriego, system ekspertowy, programowanie dynamiczne, procesy biznesowe

1. Introduction

In the last years companies started to implement IT governance ideas and to build their own corporate architecture. The corporate architecture not only contains the technical infrastructure, software solutions and data models, also the business architecture is a one of its components. Companies invest in the repositories of business processes, where hundreds of

models are being stored. Medium size government office or a company can have more than 100 processes, each containing around 10-30 tasks.

Those models are then used to optimise the processes to shorten the response time, to minimise the costs, to maximise the turnover. Those processes should also be revised when the market or the company changes and this revision can be partly done by the machine.

Below you will find the idea of automatic optimisation/scheduling tool based on Petri Nets model of the process, but it can be easily applied also for a BPMN (Business Process Modeling Notation) model or any other notation which uses the concept of a task.

The main advantage of this method is the ability to revise only those parts of the model, to which the modification was applied. Recalculating everything can be avoided, while making the change in the business process.

2. Method with simulation based genetic optimiser

2.1. Petri Nets as a model of production process

Building the process model, nevertheless if it's technological or organizational is based on the assumption that it can be divided into tasks.

Therefore business processes are sets of tasks, such as: materials storing, goods relocation, parts combining, functionality tests, client request verification etc. Each task needs time to be prepared and time to be finished. There are also expenses associated with the task, salary, costs of workplace, materials, tools depreciation. Together with times and costs of execution and preparation, each particular job is characterised by its inputs and outputs. By inputs we can understand all resources, by outputs the results of the job and in some cases returned resources.

The image below (Fig. 1) presents an example of the task modelled using Petri Nets. It has two transitions, first one simulates the start and the second the end of the task. Input places of the first transitions plays part of resources, place between the transitions describes the status of the task, the last place is an equivalent of the result.

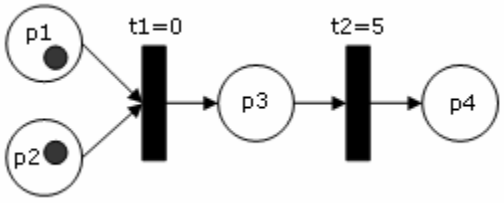

Fig. 1. Example of a task, with two different resources and one product
Rys. 1. Przykład zadania wymagającego dwóch typów zasobów

Resources needed as well as products are represented by tokens. In this not-coloured Petri Net for each type of element the separate place is used. Therefore, if the job needs two types of parts it has two input places.

The process is started by an event, internal or coming from the market – outside of the company. Events are modelled by input place, where the simulator can add tokens (request from the customer, resources from suppliers, information from the sales department etc.).

2.2. Optimiser

For optimisation the genetic algorithm with gene mutation and single point crossing was used. Even in the simplest form genetic algorithms are rather independent from the starting point and they can be used together with simulation results, when the goal function is not known or difficult to define.

The optimisation problem is represented as a vector of task start transitions firing delays.

Genotypes, from the genetic algorithm, are applied to the model, creating a phenotype. The data gathered during the simulation experiment is employed to estimate the value of the fitness function, which is obviously used to select next population of genotypes. The idea of simulation-based optimiser, presented by Peter Köchel [1], is described on a figure below (Fig. 2).

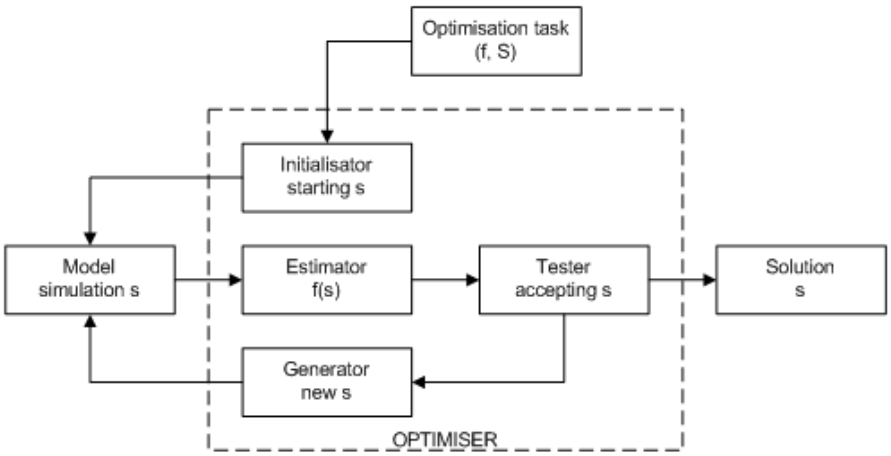

Fig. 2. Optimiser using simulation as an input for estimation of the fitness function value
Rys. 2. Optymalizator używający wyników symulacji do estymacji wartości funkcji przystosowania

More detailed description of this method can be found in the earlier publication [2].

3. Application of the dynamic programming approach

Experiments [2] show that the genetic algorithm together with a simulator allows finding best model parameters (transition delays) to optimise chosen fitness function. It is obviously simple, but it is also time consuming. It takes a lot of operations to solve the problem – to create next generation.

In this chapter the idea of using one of the Petri Nets features – scalability and the concept of dynamic programming to redesign previously described method is presented.

3.1. Scalability

In a business process, elements like warehouses, factories, planes or ships can be found. Furthermore the same process can be analysed on a level of particular machines working on a production line or web services updating the database records. Model should be then very flexible when it comes to scalability, to be able to represent processes on any level of complexity.

Thanks to Petri Nets, the process model can be divided into subprocesses or merged. From one point of view the whole engagement can be treated as a one task, while from another it can be complicated, multi-element process, just like on an example below (Fig. 3).

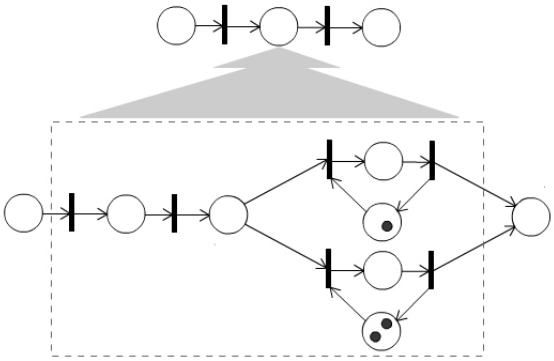

Fig. 3. Different complexity levels
Rys. 3. Różne poziomy szczegółowości

This flexibility is not only useful for presentation reasons, but it also complies with all the requirements which must be fulfilled in case of the dynamic programming. The optimisation task in the form of a business process model can be divided into smaller sub-problems and their solutions can be merged into the overall solution.

3.2. Dynamic programming procedure

The procedure proposed in this paper consists of the following steps:

A. Find in business process model P subprocesses P1-Pn; in process P replace all subprocesses by single tasks to get the P'.
B. For each subprocess Pi in P1-Pn, if it is still has more tasks then set threshold, go through step A with P = Pi.
C. For each process Pi, which does not contain any subprocesses find the set of non-dominated solutions Si (see point 3.2.2).
D. For each process Pi that consists of activities and subprocesses, for which solutions S has been already generated, find the the set of non-dominated solutions Si. For simple activities use its attributes (i.e. cost, time constraints etc.) for estimation of fitness functions f. In case of tasks that are subprocesses replacements apply generated solutions.
E. Repeat step D until the solution S for the original business process P is reached.

3.2.1. Dividing the problem

To be able to use the dynamic programming approach we need to be able to divide the problem. In many cases business processes are modelled as a high level – main process containing activities and subprocesses defining those activities. Then this part of the job is already done.

In other cases there are several automatic methods which can be applied. Petri net is a graph, therefore it can be analysed like any other graph.

Algorithm should find in a graph sub-graphs that:

A. Are composed of whole tasks. Tasks represented in a model by two transitions joined with a place cannot be told apart.
B. Starts with one place or one transition (task starting transition) and ends as well with one place or transition (task end transition). This will allow replacing it by a single task.
C. Incorporate a synchronisation of the flow for any flow fork it contains.

3.2.2. Finding the solution for the subprocess

To find a solution S_i to process P_i, which does not contain any subprocesses genetic algorithm together with simulator, described in the chapter 2, can be used.

In a model time constraints, costs, profits and other necessary attributes are assigned to places and transitions. Data gathered during the simulation is used to estimate a value of a fitness function for each objective.

Genetic algorithms are used to find the solution – a genotype, which minimise the fitness function. Even if there is more than one objective and the fitness function is a vector of functions (Eq. 1) it is in many cases changed to scalar using a vector of weights.

$$F(s) = [f_1(s),...,f_n(s)] \tag{1}$$

In this method we need to find a set of non-dominated solutions - individuals that will next be used for solving the higher level problem. The solution s is not dominated when it cannot improve any objective without degrading at least one another:

$$\forall s_n \neq s : \exists x : f_x(s) < f_x(s_n) \tag{2}$$

This non-dominated set is called the Pareto front. Searching for this kind of set affects the fitness function used for the tournament selection method. Selection drives the population towards better solutions. In this case better individual means:

a) not dominated,
b) dominating more individuals, like i.e. Strength Pareto Evolutionary Algorithm [3, 4];
c) lying further from the existing not dominated individuals, like i.e. Neighborhood Search Genetic Algorithm by Dias and de Vasconcelos [5].

The output from the optimisation will be a Pareto front S_i describing a set of optimal solutions s_i. Such a solution s_i is composed of a vector of transitions firing delays – a genotype together with estimated values of fitness functions vector $F(s_i)$.

3.2.3. Merging the solutions

Obviously choosing the best solutions for subprocesses and joining them together not necessarily gives the best overall solution. It might work fine when we have only one objective and it is time, then the faster task is done the better. For the solution on a higher level, for the process that contains subprocesses, the same optimiser can be applied with one modification. The genotype should be composed not only of start transitions delays but also end transitions delays in case of tasks being replacements of subprocesses.

The Pareto front for the subprocess S_i, found in a previous iteration will be used to evaluate the task, which represents this subprocess P_i in this iteration. The genetic algorithm generates starting and ending transitions delays. Knowing the duration of the subprocess and having the S_i, the particular s_i and the $F(s_i)$ can be found and used in the selection tournament method.

As an effect of the merging (optimising higher level process) the Pareto front with optimal solutions is being defined. It can be a final solution or an input for the next iteration, just like on a diagram (Fig. 4).

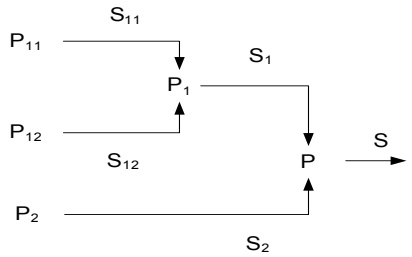

Fig. 4. Solution merging schema
Rys. 4. Schemat łączenia rozwiązań

4. Conclusion

Experiments described in earlier work [2] show that the genetic algorithm together with a simulator is extremely useful in the field of continuous parameter optimisation. It can be used to solve variety of problems, due to its robustness and its ability of escaping from local optima.

The method described in this paper is a possible solution for continuous improvement and adaptation to new conditions, which occur on the market and inside each company almost every day. The advantage of this procedure is that it allows analysing and applying the innovation without recalculating everything from the beginning. Implementing the change causes revision of only those subprocesses where the transformation takes place.

BIBLIOGRAPHY

1. Köchel P.: Solving logistic problems through simulation and evolution. Proceedings of the 7th International Symposium on Operational Research, Ljubljana 2003.
2. Wibig M.: Optimization of Projects Logistics Processes Using Petri Nets. Proceedings of XII System Modelling and Control Conference in Polish Journal of Environmental Studies, 2008.
3. Fieldsend J. E., Everson R. M. and Singh S.: Extensions to the Strength Pareto Evolutionary Algorithm. IEEE Transactions on Evolutionary Computation Vol. 2001, 2001.
4. Zitzler E.: Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. Swiss Federal Institute of Technology, Zurich 1999.
5. Dias A. H. F. and de Vasconcelos J. A.: Multiobjective Genetic Algorithms Applied to Solve Optimization Problems. IEEE Transactions on Magnetics, Vol. 38, No. 2, 2002.
6. Carmen T. H., Leiserson C. E., Rivest R. L. and Stein C.: Introduction to Algorithms (2 ed.). Mit Press and McGraw-Hill, 2002.
7. Bellman R. E.: Dynamic Programming. Dover Publications 2003.
8. Macias E. J. and de la Parte, M. M. P.: Simulation and optimization of logistic and production systems using discrete and continuous Petri nets. Simulation, Vol. 80, 2005.
9. van Veldhuizen D. and Lamont G.: Multiobjective Evolutionary Algorithms: Analyzing the State-of-Art. Evolutionary Computation, Vol. 8, 2000.
10. Fonseca C. M. and Fleming P. J.: An Overview of Evolutionary Algorithms in Multiobjective Optimisation. Evolutionary Computation, Vol. 3, 1995.

Recenzenci: Dr inż. Henryk Josiński
Dr hab. inż. Andrzej Kwiecień, prof. Pol. Śląskiej

Wpłynęło do Redakcji 31 stycznia 2010 r.

Omówienie

Niniejszy artykuł przedstawia koncepcję wykorzystania skalowalności modelu opartego na sieci Petriego oraz idei pochodzącej z programowania dynamicznego do optymalizacji procesów biznesowych.

Zgodnie z pomysłem, z takiego procesu można wydzielić podprocesy, by zmniejszyć jego złożoność. Taki podział opiera się na zdolności sieci Petriego do prezentowania i analizy problemu z dowolnym poziomem szczegółowości (rys. 3). Wydzielanie podprocesów polega na znalezieniu takiego fragmentu grafu P, który zawiera ustaloną liczbę całych zadań oraz leży pomiędzy dwoma tranzycjami lub miejscami i dla wszystkich elementów rozdzielających przepływ zawiera odpowiadające im elementy synchronizujące (punkt 3.2.1).

W kolejnym etapie dokonuje się optymalizacji procesów cząstkowych i wykorzystuje jej wyniki do analizy uproszczonego procesu nadrzędnego (rozdział 3.2.3).

Celem algorytmu optymalizującego jest wyznaczenie zbioru rozwiązań niezdominowanych – frontu Pareto (rozdział 3.2.2). Na jego podstawie dokonywane jest oszacowanie wartości funkcji przystosowania podczas działania algorytmu dla procesu nadrzędnego lub w przypadku głównego procesu stanowi on rozwiązanie zadania optymalizacyjnego.

Algorytmy genetyczne doskonale radzą sobie z różnego rodzaju problemami dzięki swojej stabilności i odporności na optima lokalne. Zaletą opisanego powyżej podejścia jest to, że aby dostosować proces do nowych warunków lub wprowadzić do niego innowację, nie jest konieczne wykonanie całości obliczeń od początku. Wprowadzenie zmiany wymaga jedynie analizy tych elementów, których ta zmiana dotyczy.

Address

Mateusz WIBIG: Deloitte Business Consulting, Technology Integration, al. Jana Pawła II 19, 00-854 Warszawa, Poland, mwibig@deloittece.com .