Optimizing Events Traffic in Event-based Systems by means of Evolutionary Algorithms

Jiri Kubalik
Institute of Software Technology
and Interactive Systems
Technical University in Vienna
Favoritenstr. 9/188, A-1040 Vienna, Austria
kubalik@labe.felk.cvut.cz

Richard Mordinyi
Space-Based Computing Group
Institute of Computer Languages
Vienna University of Technology
Argentinierstr. 8, A-1040 Vienna, Austria
richard@complang.tuwien.ac.at

Abstract

This paper presents a new approach for solving network flow optimization problems. In particular, the goal is to optimize the traffic in the network structured event-driven systems as well as to provide means for efficient adaptation of the system to changes in the environment - i.e. when some nodes and/or links fail. Many network flow optimization problems belong to the class of NP hard problems, which can only be solved by using some heuristic approach. In this paper we describe an application of a recently introduced iterative optimization algorithm with evolved improvement steps. This algorithm is well suited for solving hard discrete combinatorial problems as well as rescheduling-like problems; thus it fits the addressed adaptability issue very well.

1 Introduction

Event-based systems [8] are increasingly used in business and industrial practice [10]. Many safety-critical event-based systems are connected with point-to-point data links that have good performance but poor flexibility. Networks that connect these event-based systems are much more flexible, but introduce new challenges of routing, performance for exchanging event messages (e.g., bottlenecks may be created that unnecessarily limit the maximal load the network can transport), fault tolerance, and the security of such messages. In an event-based system interaction between components is done by generating and receiving event-notifications, where an event is any occurrence of a happening of interest [10]. The event messages to be exchanged in the network in a given timeframe can be modeled as a network flow optimization problem to investigate the feasibility of routing a given set of messages; different types of messages can model the different levels of priority and security. In [10] there are several routing techniques listed pointing out in how many different ways notifications between publisher and subscriber can be routed. This paper should be a contribution to those algorithms trying to find an optimal route from publisher to subscriber.

Many network flow optimization problems belong to the class of NP hard problems for which no effective algorithms that would find optimum solution exists. Thus, heuristic approaches, which can deliver acceptably good solutions in reasonable computational time, have been widely studied and developed for those problems. Typical representative of those approaches are approximation techniques, tabu search techniques, ant colony based optimization techniques and evolutionary algorithms. In [11], Pioro and Gajowniczek implemented a stochastic approach called Simulated Allocation algorithm for solving capacitated and uncapacitated multicommodity integral flow allocation problems. Tabu search-based approaches for solving the capacitated multicommodity network design problems have been proposed in [14] and [4]. An example of an application of the ant colony optimization algorithm to an integral multicommodity flow assignment problem can be found in [13]. The ant algorithm is used there to minimize the overall network flow - the function defined as the sum of the link flows of all the links in the network. Traditional Lagrangian relaxation and sub-gradient optimization methods were used for solving route selection and capacity and flow assignment problem in communication networks in [5] and [12]. For the same problems an evolutionary algorithm called Network Genetic Algorithm has been proposed in [7]. It uses a special solution encoding and employs a parallel evolu-

Second International Conference on Availability, Reliability and Security (ARES'07)
0-7695-2775-2/07 $20.00 © 2007 IEEE

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: Universitatsbibliothek der TU Wien. Downloaded on December 17, 2009 at 08:31 from IEEE Xplore. Restrictions apply.

tion approach using dynamically changing genetic parameters. In [1], Arabas and Kozdrowski describe an application of evolutionary computation to telecommunication network design. It focuses on the network dimensioning problem in which the network structure is given and the task is to find values of link capacities that are optimal according to a certain criterion.

Up to now not many approaches to network design and network flow optimization problems based on evolutionary algorithms (EA) can be found in the literature. This is because the network flow optimization problems impose a strong constraint on the flow balance at each node of the network, i.e. the total node's output must be equal to the total node's input. And it is very hard to either design an EA such that it would generate only feasible solutions (with balanced flows at all nodes) or to support the EA with such a repair algorithm that could efficiently transform each illegal solution to the legal one. None of the above options is usually the case so a penalty must be used then to guide the EA towards the feasible solutions. However, the penalty approach greatly decreases the quality of the evolved solutions and it slows down the convergence towards the final solutions as well. In [9], Munakata and Hashier applied an EA to the maximum flow problem. They introduce an energy level assigned to each node, which is zero if the node is balanced and has its flow at the maximum capacity. Such a node is considered stable. Otherwise, the node's energy increases accordingly as the level of its flow unbalance increases and its flow decreases and is unstable. Genetic operators used in this approach prefer more stable nodes so that the evolution process is biased towards solutions with stable nodes. They showed their genetic algorithm can find optimal or near optimal solutions for simple networks. However, the balancing objective still remains an issue as it makes the algorithm to converge slowly to the desired solution.

In [3], Bramlette proposes an indirect representation and a two-phase optimization method to overcome the difficulties with satisfying the constraint of balanced flows when solving the maximum flow problem. Here, the chromosome is a list of n integer variables, where n is the number of flow allocations. The trick is that the variables do not represent the direct flow allocations. Instead, they represent only desired relative sizes. When the fitness of a chromosome is to be assessed the actual flows on output links of every node are calculated first so that they respect i) the capacity constraints and ii) a relaxed inequality constraint I1 that no node's total output exceeds its total input. It is important that any set of n integer values (i.e. any chromosome) determines a valid set of actual flow allocations. In the first step of the optimization algorithm, the genetic algorithm evolves solutions according to the maximal network output criterion. Then, the best solution achieved is further refined to have all nodes' flows balanced. This is achieved by a deterministic algorithm that removes the excess input at each node, returning the flow upstream to the network's source node. It was shown that the proposed approach is very efficient as it guarantees that only legal solutions are generated during the genetic algorithm's run, so no penalty functions have to be used. On the other hand, a very limiting restriction of this approach is that it can be used only for acyclic networks.

In this paper we present a new efficient evolutionary-based approach for solving network flow optimization problems. We introduce a direct representation with a repair algorithm, which ensures that only legal solutions are generated by genetic operators during the course of the optimization process. The repair algorithm takes the inspiration from the indirect representation proposed for acyclic graphs by Bramlette in [3] and extends it for the networks with cycles as well. Further, we use an Iterative **P**rototype **O**ptimisation with **E**volved I**M**provement **S**teps (POEMS) [6] as the optimization framework. The POEMS algorithm has already proved to perform well on hard binary string optimization problems, traveling salesman problem, and the real-valued parameter optimization problems. This paper reveals some interesting aspects of using the algorithm and shows that the POEMS algorithm can be used for effectively solving hard network flow optimization problems as well. The proposed approach is experimentally evaluated on cyclic and acyclic networks and its performance is compared with other approaches found in the literature as well as with the standard genetic algorithm.

Here, we focus on the maximum flow problem with capacities and flows being integer. For an experimental evaluation of our approach we use the original networks introduced in [9] as well as two extended networks. This problem was chosen to show capabilities of our proposed approach to solve network flow optimization problems, in which the balanced flow objective plays a crucial role.

The paper is structured as follows. Next two sections describe the proposed approach and the implementation issues. Sections 4 and 5 describe the test problems and the experimental setup used in the experiments. Section 6 presents the achieved results. The last section concludes the paper with a list of topics for our further research.

2 Proposed Approach

The problem that we deal with in this work is defined on a directed graph with a set V of m nodes and a set E of n directed edges connecting the nodes. Each edge is assigned its integer capacity. The goal is to maximize the total flow from the source to the sink node while satisfying two conditions - 1) the flow at each edge must be an integer between zero and its flow capacity and 2) at each node, the incoming flow and outgoing flow must be in balance.

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: Universitatsbibliothek der TU Wien. Downloaded on December 17, 2009 at 08:31 from IEEE Xplore. Restrictions apply.

Our approach takes the inspiration from [3]. Here, a repair algorithm is also used that guarantees that any candidate solution generated by the genetic operators of the genetic algorithm can be transformed to the one, which complies with the constraint that no node's total output flow exceeds its total input flow. Moreover, it extends for networks with cycles as well, which significantly extends a scope of possible applications.

The solutions to the maximum flow problem are again represented by a linear chromosome as a list n integer variables, where n is the number of flow allocations. Unlike the Bramlette's approach, the variables represent desired flow allocation (not the relative sizes) so that the value of each variable ranges from 0 to the flow capacity of the respective edge. The genetic algorithm searchs for an optimal solution with maximal flow from the source to the sink node while considering only the relaxed node's balance flow constraint.

The repair algorithm that is invoked to adjust the flow allocations to the valid ones that already satisfy the relaxed flow balance constraint is shown in Fig. 1. A set S of nodes to be processed is maintained. At the begining it sets S to contain all nodes in the graph $S = V$. Then, it iterates in the main loop, processing nodes from S one by one, until the S is empty. In each iteration it takes the first node from S, recalculates its output flow allocations so that the total node's input flow is distributed to its outgoing edges proportionally to the desired flow allocation sizes, while satisfying the capacity constraints imposed on the outgoing edges. Thus, not more than the total input flow can be assigned to the output flows. If any of the node's output edges is assigned smaller flow than was the original one, then all nodes to which the edge leads are inserted to the set S. This is because as the total input flow of the corresponding nodes decreases the flow allocations at their outgoing edges might have to be re-adjusted. This procedure provides a means for finding valid flow allocations with respect to the relaxed balance constraint in a finite number of iterations. Let us assume that the graph contains a group of nodes that affect mutually each other (they constitute a cycle) so that reduction of the output flow of one node causes that some other nodes of the group have to be recalculated. Then the recalculation of those nodes will likely take more iterations, where some of the nodes will be recalculated multiple times. However, this process will end up either when all the nodes get to the state with the balanced flow or when the edges belonging to the cycle are assigned zero flow (due to the fact that the output flow of the processed node can only decrease), so that no further node's recalculations can be invoked along such an edge any more.

The optimal solution found in the first phase is then finalized by a deterministic procedure that adjusts flows in order to achieve the precise balance of the input-output flow at each node. A outline of this procedure is shown in Fig. 2.

input:	Original flows stored in the chromosome
output:	Adjusted flows that satisfy the condition that in each inner node of the network the sum of its output flows is less or equal to the sum of its input flows.
1.1	Assign original flows to network edges
1.2	Initialize the list of nodes to be processed $S = V$
2	do
3	Take the first node v of the list S
4.1	Recalculate output flows of node v
4.2	Add all nodes that have been affected by this action to list S
5	while($S \neq \{\}$)

Figure 1. Repair algorithm for direct representation.

For any node that has the total input flow bigger than its total output flow it finds a path from the given node to the source node, and then decreases the flow along that path as much as possible. This might be repeated several times for one node, until the excess input flow has been completely removed.

In the first phase of the optimization process any evolutionary algorithm can be used to find the maximum flow solution. Note, that this is an integer-valued parameter optimization problem of n parameters, which might be a hard task for standard genetic algorithms for larger problem sizes. In this work, we propose an implementation of a new evolutionary-based approach called Iterative **P**rototype **O**ptimisation with **E**volved I**M**provement **S**teps (POEMS) proposed in [6].

2.1 POEMS

Typically, an evolutionary algorithm (EA) evolves a population of individuals, where each individual encodes a complete solution to the problem at hand (a complete set of problem control parameters, a complete schedule in JSP, a complete tour for TSP, etc.). For large problem instances such EA often fails to find the optimal or any well-fit solution at all.

POEMS is an iterative optimization framework, where the evolutionary algorithm (EA) does not handle the complete candidate solution to the problem at hand. The main idea behind POEMS (see Figure 3) is that some initial prototype solution is further improved in an iterative process, where the most suitable modification of the current prototype is evolved by means of the EA in each iteration. The modifications are represented as a sequence of primitive actions/operations, defined specifically for the problem at hand. The evaluation of action sequences is based on how

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: Universitatsbibliothek der TU Wien. Downloaded on December 17, 2009 at 08:31 from IEEE Xplore. Restrictions apply.

input:	Flows that already satisfy the condition that in each inner node the sum of its output flows is at most as big as the sum of its input flows.
output:	Balanced flows for which the balance condition holds that every inner node has its total output flow equal to its total input flow.
1	while(exists node v with unbalanced in-out flows)
2	$flow\_diff = v.in - v.out$
3	do
4.1	Find an acyclic path P from v to the source node s such that all edges in the path are assigned a positive flow.
4.2	Set $reducible$ to the flow of the edge with the minimal flow along the path P
4.3	$reducible = min(reducible, flow\_diff)$
5	Decrease flows of all edges of path P by value of $reducible$
6	$flow\_diff = flow\_diff - reducible$
7	while($flow\_diff > 0$)

Figure 2. Algorithm for finding balanced flows in every inner node of the network.

good/bad they modify the current prototype, which represents an input parameter of the EA. Sequences that do not change the prototype at all are penalized in order to eliminate generating trivial solutions. After the EA finishes, it is checked whether the best evolved sequence improves the current prototype or not. If an improvement is found, then the sequence is applied to the current prototype and the resulting solution becomes the new prototype. Otherwise the current prototype remains unchanged for the next iteration.

Representation. The EA evolves linear chromosomes of length *MaxGenes*, where each gene represents an instance of certain action chosen from the set of elementary actions defined for the given problem. Each action is represented by a record, with an attribute *action_type* followed by parameters of the action. Besides actions that truly modify the prototype there is also a special type of action called *nop* (no operation). Actions with $action\_type$=*nop* are interpreted as void actions with no effect on the prototype, regardless of the values of their parameters. A chromosome can contain one or more instances of the *nop* operation. In this way the variable effective length of chromosomes is implemented. An important aspect of this implementation is that any temporarily inactivated action can be activated again later on (with its formerly evolved parameters) just by switching its $action\_type$ on.

Operators. The representation allows to use a variety of possible recombination and mutation operators such as standard 1-point or 2-point crossover and a simple mutation.

```
1  generate(Prototype)
2  repeat
3      BestSequence ← run_EA(Prototype)
4      Candidate ← apply_to(BestSequence, Prototype)
5      if(Candidate is_better_than Prototype)
6          Prototype ← Candidate
7  until(POEMS termination condition)
8  return Prototype
```

Figure 3. An outline of POEMS algorithm.

Evolutionary model. The design and configuration of the EA can differ for each particular optimization problem. In general, the EA is expected to be executed many times during the whole run of the POEMS. Thus, it must be configured to converge fast in order to get the optimized action sequence in short time. As the EA is evolving sequences of actions to improve the solution prototype, not the complete solution, the maximal length of chromosomes $MaxGenes$ is typically shorter than the actual size of the problem. The relaxed requirement on the expected EA output and the small size of evolved chromosomes enables to setup the EA so that it converges within a few generations.

It is important to note, that POEMS is not optimizing the prototype via improvement steps that are purely local with respect to the current prototype. In fact, long phenotypical as well as genotypical distances between the prototype and its modification can be observed if the system possesses a sufficient explorative ability, see [6]. The space of possible modifications of the current prototype is determined by the set of elementary actions and the maximum allowed length of evolved action sequences. The less explorative actions are and the shorter sequences are allowed the more the system searches in a prototype neighborhood only and the more it is prone to get stuck in a local optimum, and vice versa.

3 Implementation Issues

POEMS is a general optimization framework. The problem specific part is the implementation of the engaged EA. One has to design the representation of the evolved action sequences, genetic operators operating on the sequences, and the evolutionary model. Perhaps, the most important is a proper choice of the set of action types. They should be chosen so that the space of possible candidate action sequences is rich enough to ensure a sufficient exploration capabilities of the whole system.

Representation. In this work, besides the nop action we have used just one effective action type $add(edge, flow\_change)$. It adds the value $flow\_change$ to the flow assigned to the respective $edge$. When initializing or mutating this action, the value of $flow\_change$

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: Universitatsbibliothek der TU Wien. Downloaded on December 17, 2009 at 08:31 from IEEE Xplore. Restrictions apply.

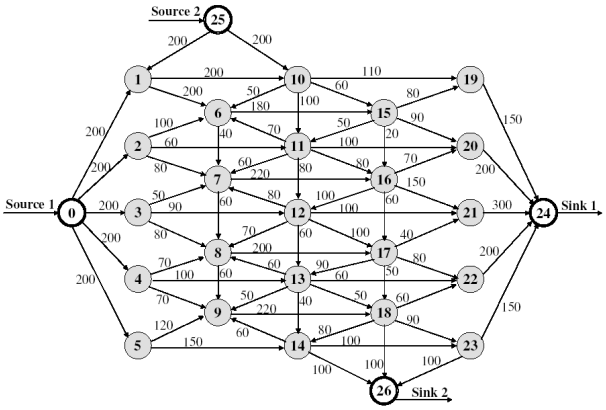

Figure 4. Graph-ext with cycles and two source and two sink nodes

is chosen randomly so that when the action is applied to the $edge$ the resulting flow must be in the interval $(0, edge.capacity)$.

The chromosome (i.e. the candidate action sequence) is represented as a list of $MaxAction$ actions (active or inactive), each of them operating on one edge of the current solution prototype. At least one of the actions in each sequence must be active with non-zero argument in order to eliminate trivial solutions (i.e. action sequences that do not modify the prototype at all).

Operators. A generalized uniform crossover proposed in [6] was used here, that forms a valid offspring as an arbitrary combination of parental genes. Both parents have the same probability of contributing its genes to the generated child, and each gene can be used only once. Mutation operator changes either the $action\_type$ (activates or inactivates the action) or the parameter $flow\_change$.

Evolutionary model. A simple steady-state EA that iteratively modifies a single population of individuals was used. In each iteration a couple of individuals is selected that undergoes the crossover operation. The generated offspring is then modified by the mutation operation and assigned the fitness. Then one of the worst individuals in the current population is chosen as the replacement. Finally, the offspring is placed on the position of the replacement in the population if it is better than the replacement. A tournament selection was used for selecting the parents.

4 Test Problems

The following graphs were used for the experimental evaluation of the proposed approach:

- Original Munakata's acyclic directed graph with one source-sink pair, referred to as Graph 1 in [9]. It has 25 nodes and 49 edges. The maximum flow is 90.
- Original Munakata's graph with one source-sink pair and cycles, referred to as Graph 2 in [9]. This graph has 25 nodes and 56 edges. The maximum flow is 91.
- Extended Munakata's Graph 2 with cycles and two source and two sink nodes and bigger capacities denoted as Graph-ext, see Fig. 4. This graph has 27 nodes and 71 edges. In order to increase the search space the capacities of edges are ten times bigger then those in the original Munakata's graph.
- Extended Munakata's Graph 2 with bidirectional inner links denoted as Graph-bidirect. This graph is the same graph as the Graph-ext but the inner edges i.e. the edges that do not lead neither to source nor to sink nodes are bidirectional. In fact, each of the inner edge is doubled with the same capacity - one edge for each direction. This adds 56 edges to the graph so it has 27 nodes and 127 edges in total.

5 Experimental Setup

Results achieved with POEMS were compared with results published in [9] and [3] and the standard genetic algorithm (SGA) using 2-point crossover, simple mutation, and tournament selection. For each experiment, 20 independent runs of POEMS and SGA have been executed and the following statistics have been recorded:

- Best - the best-of-run solution fitness.
- Mean, StDev - the mean and standard deviation of the 20 best-of-run values.

The following configuration of the EA engaged in POEMS was used in the experiments:

- Population size: 50,
- Number of fitness function evaluations: 500 (original Munakata's networks), 750 (extended networks),
- *MaxGenes*: 20 (Munakata's networks), 50 (extended networks),
- $P_{crossover} = 0.9$, $P_{mutation} = 0.1$.

The following configuration of the SGA was used:

- Population size: 50,
- $P_{crossover} = 0.9$, $P_{mutation} = 0.015$.

The total number of fitness function evaluations of both the SGA and the POEMS was set to 10000 (original Munakata's networks), and 20000 (extended networks).

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: Universitatsbibliothek der TU Wien. Downloaded on December 17, 2009 at 08:31 from IEEE Xplore. Restrictions apply.

Table 1. Experimental results on original Munakata's Graph 1

Algorithm	Best	Mean	StDev
Munakata	90.0 (3400)	90.0	0.0
Bramlette	90.0 (1450)	90	0.0
SGA-direct	80.2	79.4	0.49
POEMS-direct	80.8	80.07	0.45
POEMS-repair	90.0 (3100)	90	0.0

Table 2. Experimental results on original Munakata's Graph 2

Algorithm	Best	Mean	StDev
Munakata	89.0	85.90	2.20
SGA-direct	88.7	87.93	0.40
SGA-repair	91.0	90.63	0.67
POEMS-direct	88.9	88.29	0.39
POEMS-repair	91.0	90.65	0.81

6 Experimental Results

Tables 1 and 2 show results achieved on the original Munakata's Graph 1 and Graph 2. The tables compare results presented in [9] and [3] with our results achieved with SGA and POEMS using a direct representation with a penalty fitness function, and POEMS using the repair procedure proposed in this paper. It shows that except the SGA-direct and POEMS-direct, all other approaches were more or less successful on those test graphs. SGA-direct and POEMS-direct were not able to find a solution with the balanced flow in the given time in any out of the 20 runs. It also shows that our proposed repair procedure in combination with both the SGA and POEMS clearly outperforms the Munakata's results on the Graph 2.

Tables 3 and 4 compare the SGA using the repair procedure with two versions of POEMS using the repair procedure - POEMS-repair (random) generates the starting prototype by randomly sampling the flow allocations, POEMS-repair (empty) starts with empty solutions (all flows are zero). The Bramlette's approach can not be used because both Graph-ext and Graph-bidirect are graphs with cycles.

Table 3. Experimental results on Graph-ext

Algorithm	Best	Mean	StDev
SGA-repair	1187.0	1177.90	5.28
POEMS-repair (random)	1190.0	1189.45	0.60
POEMS-repair (empty)	1190.0	1189.45	0.99

Table 4. Experimental results on Graph-bidirect

Algorithm	Best	Mean	StDev
SGA-repair	1206.0	1186.95	8.65
POEMS-repair (random)	1230.0	1210.75	19.21
POEMS-repair (empty)	1230.0	1211.85	22.93

The Munakata's approach were not implemented but it can be expected that it would not perform very well anyway as it already had some problems when solving maximum flow problem on much simpler Graph 2. The results show that POEMS-repair already outperforms the SGA-repair algorithm. This is in agreement with our expectations that as the size of the problem increases (both the number of edges and their capacities increased from the original Graph 1 and 2) the performance of the standard evolutionary approach decreases. Also an interesting observation is that POEMS starting from the empty solution prototype achieves as good solutions as when it starts from a randomly generated starting prototype of non-zero fitness. This means that this approach is capable of evolving a good solution to the problem from scratch. This is particularly useful when no effective heuristic can be used to generate the starting feasible solution. Then, the POEMS can start at least from the empty solution, which is usually a feasible one.

7 Discussion and Conclusions

This paper presents a new evolutionary-based approach for solving network flow optimization problems. An efficient repair procedure was proposed that guarantees that only valid solutions are processed during the optimization process. Further, an application of the iterative optimization framework called POEMS was proposed for the maximum flow optimization problem. First experiments are very promising. They show that this approach outperforms other evolutionary-based approaches developed for the maximum flow problem and outperforms also a standard genetic algorithm using the same repair procedure.

Our primary interest is in the optimization of events traffic in event-based systems. Below is a list of variations of the original maximum flow problem that we believe are in principle solvable by the proposed iterative-evolutionary approach utilizing the repair procedure.

Simple Network Problem. Find an optimal solution to route a set of events transports through a network with limited capacity (e.g., in a given timeframe). Links between nodes can be directed or undirected (in the latter case each undirected link would just be represented by two links of the same capacity, each for one direction), and can have other

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: Universitatsbibliothek der TU Wien. Downloaded on December 17, 2009 at 08:31 from IEEE Xplore. Restrictions apply.

attributes like cost (fixed/variable) etc. Further, a list of desired transports (each transport has its pair of source-sink nodes, and the number of units to be transported) is given. The following goals can be defined:

- To find a feasible solution.
- To find a minimal cost feasible solution.
- If no feasible solution found then give a list of limiting bottleneck links.

Advanced Network Problems (Networks with Tags). Messages, links, and nodes can have (combinations of) tags attached, e.g., to denote transport priorities, reliability of lines, right of a transport to use a given link. A message with a tag may travel only nodes and links that also have this tag. The goals would be the same as in the simple network problem above.

Networks with Node/Link failures. During network operation, nodes or links may be unavailable due to a node or link failure. Solve the simple network problem, if any one of the nodes or links becomes unavailable. If possible, use the solution of the problem without failure to derive the solution for the corresponding network with a single failure faster than calculating the solution anew in every new combination. This problem scenario seems to be particularly suitable for the POEMS framework because in POEMS a sequence of simple actions is sought such that it improves some current prototype solution to the problem. So, taking the solution to the problem without failure as the initial prototype, POEMS could be used to find such a sequence of actions that would make the solution valid in the new context with the node/link failure.

The investigation of the possible applications of the presented approach to the problems listed above is a subject of our further research.

Acknowledgments

This research work has been supported by a Marie Curie Transfer of Knowledge Fellowship of the European Community's 6th Framework Programme under the contract MTKD-CT-2005-029755.

References

[1] Arabas J. Kozdrowski S., "Applying an Evolutionary Algorithm to Telecommunication Network Design", *IEEE Transactions on Evolutionary Computation*, Vol. 5, No. 4, 2001.

[2] Birman, K. P., "Reliable Distributed Systems", Springer New York, 2005.

[3] Bramlette M. F., "Finding Maximum Flow with Random and Genetic Search", *Proc. First IEEE Conference on Evolutionary Computation, Orlando, Florida*, pp. 296-299, 1994.

[4] Crainic T.G., Gendreau M., Farvolden J., "A Simplex-Based Tabu Search Method for Capacitated Network Design", *INFORMS Journal on Computing*, Vol. 12, No. 3, pp. 223-236, 2000.

[5] Gavish B., "A system for routing and capacity assignment in computer communication networks", *IEEE Transactions on Communications*, Vol. 34, No. 4, pp. 360-366, 1989.

[6] Kubalik J. and Faigl J., "Iterative Prototype Optimisation with Evolved Improvement Steps", P. Collet, M. Tomassini, M. Ebner, A. Ekart and S. Gustafson (Eds.), *Proc. Genetic Programming. Proceedings of the 9th European Conference, EuroGP 2006*, To be published.

[7] Lin X., Kwok Y. Lau V., "A genetic algorithm based approach to route selection and capacity flow assignment", *Computer Communications*, Vol. 26, pp. 961-974, 2003.

[8] Luckham, D., "The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems", Addison-Wesley Professional, 1st edition, 2002.

[9] Munakata T., Hashier D., "A genetic algorithm applied to the maximum flow problem", *Proc. Fifth International Conference on Genetic Algorithms, Morgan Kaufmann, Urbana-Champaign, IL*, pp. 488-493, 1993.

[10] Muhl, G., Fiege, L., Pietzuch, P., "Distributed Event-based Systems", Springer, Berlin, 2006.

[11] Pioro M. and Gajowniczek P., "Solving multicommodity interal flow problems by simulated allocation", *Telecommunication Systems*, Vol. 7, pp. 17-28, 1997.

[12] Pirkul H., "Routing and capacity assignment in backbone communication networks", *Computer Operations Research*, Vol. 24, No. 3, pp. 275-287, 1997.

[13] Walkowiak K., "Ant Algorithm for Flow Assignment in Connection-Oriented Networks", *Int. J. Appl. Math. Comput. Sci.*, Vol. 15, No. 2, pp. 205-220, 2005.

[14] Zaleta N. and Socarras A., "Tabu Search-based algorithm for Capacitated Multicommodity Network Design Problem", *Proc. 14th IEEE International Conference on Electronics, Communications and Computers*, 2004.

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: Universitatsbibliothek der TU Wien. Downloaded on December 17, 2009 at 08:31 from IEEE Xplore. Restrictions apply.