Heuristic Algorithms for the Robust PNS Problem

Dénes Almási, Csanád Imreh, Tamás Kovács

University of Szeged, Árpád tér 2, 6720 Szeged, Hungary
denes@rudanium.org, cimreh@inf.u-szeged.hu, tkovacs@inf.u-szeged.hu

József Tick

Óbuda University, Bécsi út 96/b, 1034 Budapest, Hungary, tick@uni-obuda.hu

Abstract: In this paper we study the robust PNS problem, which is the extension of the structural PNS problem used to model process network synthesis. This problem is NP-hard thus a heuristic algorithm can be very useful for large instances, where we do not have enough time for an exponential time algorithm, presenting a surely optimal solution, or they can be used to fasten branch and bound based algorithms. We present new heuristic algorithms for the solution of the problem which are extensions of the heuristic algorithms used to solve the classical structural PNS problem. The algorithms are analyzed empirically, where we compare the efficiency on randomly generated inputs.

Keywords: PNS problem; robust problems; heuristic algorithm, P-graphs; Business Process Modeling

1 Introduction

In the Process Network Synthesis (PNS for short) problem a set of materials is given and also operating units which are transforming some subset of materials into some other subset. The subsets assigned to the operating unit are called its input and output materials. In the problem two subsets of the materials are distinguished, one is the set of the raw materials and the other is the set of the desired products. Our goal is to find a minimal cost network of the operating units which can produce all desired products starting from the raw material. These systems can be modeled in the P-graph framework which is based on bipartite graphs. In these P-graphs we have two sets of vertices, one of them contains the possible materials, the other the operating units. The edges lead to an operating unit from its input materials and from an operating unit to its output materials.

Then the subgraphs satisfying some properties describe the feasible processes which produce the desired products from the raw materials. Thus our goal is to find the least expensive such subgraph. In the structural model the amounts of the material flows are not taken into account thus the cost of an operating unit is a constant, and the cost of a subgraph is the sum of the costs of the operating units contained in it. The foundations of PNS problem can be found in [9, 10], we will recall only the most important definitions in the next section.

The PNS problem and the P-graph technology to handle it were defined to model and solve problems in the chemical and allied industries. Later it was discovered that this terminology and the model itself can be very useful in other areas as well. The applications of P-graphs in workflow optimization are presented in [17, 18, 20]. It was also used in supply chain management; such results can be found at [1, 2, 8, 15, 22]. Moreover the P-graph framework was also used in planning building evaluation routes [11, 12].

In the structural PNS problem it is supposed that the exact cost of all operating unit is given in the input of the problem. Unfortunately in many applications this is not true. In these cases we do not have exact data there are some uncertainties in these values. There are several ways to solve this problem. In [16, 21] the fuzzy extension is presented of the PNS problem with applications on the area of workflow management. In [15] an application to supply chain management with uncertainty is presented where the P-graph is extended with the ROA (reliability of availability) value at the materials.

A further approach is the robust optimization where instead of the fixed values of the parameters we only know that they are in a given interval. Then we are minimizing the cost of the worst case of the possible values. Robust optimization is a widely studied area, one can find some overview in [3, 4]. The robust PNS problem was defined in [19]. In this model each operating unit has two different costs a nominal cost and an extended cost, and we know that at most b operating units have the extended cost; the others will have the nominal cost. The goal is to find the optimal solution for these assumptions. The definition of the model [19] presents a branch and bound based algorithm for the solution of the problem and also a polynomial solvable class of robust PNS problems.

In [6], it is proved that the structural PNS problem belongs to the complexity class of NP-hard problems, therefore it follows that the more difficult robust PNS problem is NP-hard as well. This means that we cannot expect a worst case polynomial time algorithm which surely finds the optimal solution unless P=NP. Therefore, some polynomial time heuristic algorithms which can find feasible solutions which are close to optimal can be very useful. They can be used if we have to solve large size problems in a short time. Moreover, such heuristic algorithms can be useful in accelerating Branch and Bound based algorithms. The application of a good starting solution can increase the efficiency of eliminating subsets not containing optimal solutions. In the case of the classical structural PNS problem some heuristic algorithms are studied in [5] and [7]. In [5] a greedy type algorithm is presented for the solution of the general problem which is based on the ideas of algorithms from set covering. In [7] a special class of PNS problems is studied.

In this paper we extend and analyze the heuristic algorithms from [5] into the more general robust PNS problem. In the next section we recall the basic definitions in the area of robust PNS problem which will be used later in the paper. Then in Section 3, the heuristic algorithms which are studied in the paper are presented. Section 4 contains the analysis of the algorithms. We present the results of an experimental analysis, where the algorithms are compared to randomly generated inputs.

2 Notations and Basic Definitions

The structural PNS problem can be modeled in the P-graph framework. In the P-graph (Process Graph) we have the set of the materials denoted by M, which contain two special subsets, the set of raw materials and the set of desired products denoted by R and P respectively. The problem also contains a set of possible operating units which can transform some sets of materials. The set of operating units is denoted by O. An operating unit *u* is given by two sets, *in(u)* denotes the set of the input materials *out(u)* denotes the set of output materials of the operating unit. This means that the operating unit can work in a solution structure if all of its input materials are produced and in this case it produces all of its output materials. The P-graph (Process Graph) of the problem is defined by the sets M and O. It is a directed bipartite graph where the set of vertices is M ∪ O, and have the following two sets of edges:

1) Edges which connect the input materials to their operating unit
2) Edges which connect the operating units to their output materials

Then some of the subgraphs of this P-graph describe the feasible solutions which produce the required materials from the raw materials. In [9] it is shown that a subgraph (m,o), where m and o are the subsets of M and O, represent a feasible solution if and only if the following properties called axioms are valid:

(A1) m contain all element of P

(A2) a material from m is a raw material if and only if no edge goes into it in the P-graph (m, o)

(A3) For each operating unit u from o there exists a path in the P-graph (m,o) which goes into a desired product from u

(A4) m is the union of the input and output material sets of the operating units contained in set o.

In the structural PNS problem we assign, to each operating unit u, a cost denoted by c(u) and the goal is to find a feasible solution, where the total cost of the contained operating units is minimal. In the robust version we assign two costs to the operating units: a nominal cost denoted by c(u) and an extra cost denoted by e(u). The sum c(u)+e(u) is called the extended cost of the operating unit. Furthermore, we have an a priori bound b, which means that in any solution, at most, b operating units can have the extended cost and the others have the nominal cost. We are interested in the worst case; therefore, if we consider a feasible solution of the problem then its cost will be the sum of the nominal costs of the operating units plus the sum of the b largest extra costs. We note that the set of feasible solutions, in the case of the robust problem, is the same as in the classical problem, the difference is that in the robust version we are considering a more difficult objective function.

3 The Heuristic Algorithms

First we present a framework heuristic which is the general version of the algorithms presented in [5]. Later we give several specialized version of the framework algorithm which can be used for the solution of the robust problem. The algorithm builds a solution step by step; it selects one operating unit in each iteration step. The algorithm uses three sets, the *set of the selected operating units,* the *set of the required materials* and the *set of the produced materials*. First the set of the selected operating units and the set of produced materials are empty and the set of the required materials is equal to the set of the desired products. Then, in each iteration step, we choose the operating unit minimizing some evaluation function and put it into the set of the selected operating units. We add its output materials to the set of produced materials, and delete them from the required materials set. Then each input material of the operating unit which is not raw material and not contained in the set of produced materials is put into the set of required materials. The procedure ends when the set of the required materials becomes empty. We obtain the feasible solution which contains the operating units from the set of selected operating units and the materials which are input or output materials of some of these operating units.

This algorithm, with some specialized evaluation functions, was defined for the solution of the structural PNS problem. In [5] it is proved that it always produces a feasible solution for the problem. Alternately, the proof does not use the definition of the evaluation function thus it works for arbitrary evaluation functions. Moreover, the set of feasible solutions is the same as in the classical and the robust PNS problem, thus we obtain the following statement.

Proposition 3.1 ([5]) The algorithm defined above results in a feasible solution of the robust PNS problem for any evaluation function on the operating units.

To specify the frame algorithm we have to define an evaluation function on the operating units. We will use a greedy type algorithm. On one hand we would like to choose an operating unit which produces the most elements from the set of the requested materials. On the other hand we would like to choose such operating units which have small cumulated costs, where in the cumulated costs not only the direct cost of the operating unit is estimated but also the cost of producing its input materials. Thus, we have to define a cumulated cost CU(o) for each operating unit, and then we choose the operating unit where the ratio of the cumulated cost and the number of the required material produced by the unit is minimal. The cumulated cost depends on the direct cost of the operating unit and also on the cost of its input materials. Since we consider the robust problem even the direct cost of the operating unit cannot be determined in advance, since we do not know whether a nominal or an extended cost will be calculated in the solution. We define, below, three possibilities for estimating the direct cost of the units and two possibilities for estimating the cost of the input materials, thus we will obtain six different estimations for the cumulative cost and this results in six heuristic algorithms.

In the case of the direct cost of the unit we can use the following estimations:

- Average cost: In this case, we use some weighted average of the two costs, thus $c_A(o)=\alpha c(o)+(1-\alpha)(c(o)+e(o))$ for some $0\leq\alpha\leq1$.
- Worst case cost: In this case we use the extended cost unless we already selected *b* operating units with at least as big extra cost as the actual unit has. In the latter case we use the nominal cost.
- Hybrid cost: This case is similar to the worst case cost but we use the average cost if we have not selected already *b* operating units with at least as big extra cost as in the actual unit.

We note that the average cost is a static one, in the sense that we can define it in advance and it is independent of the solution built by the heuristic algorithm. Contrarily, the worst case and hybrid costs are dynamic ones, they depend on the actual solution structure thus they cannot be calculated in advance.

To define the indirect cost resulting from the input materials, first we have to define a cost function *MA* on the materials. Here we use the same idea, which is used in the heuristic algorithms, of [5] and was defined in [13] for bounding function in a Branch and Bound algorithm. The only difference occurs when we use the costs of operating units in the calculation, then in the robust version we use the average cost of the operating units. We cannot use the other estimations here; since we have to calculate these costs in advance, thus we cannot apply the dynamic costs. We will use only cycle free P-graphs in our tests and the definition of the material costs are much easier in this case, therefore, we only recall the basic ideas of this simpler definition here. One can find the detailed general construction of the cost function on materials in [13].

We use two sets of materials: I denotes the materials with the given *MA* values and J denotes the complement set where we still have to calculate the value of *MA*. At the beginning, I contains the raw materials with $MA(m)=0$, and later in each step one element is moved from J to I.

We always choose such material m from J, which is only produced by operating units having all input materials in I. Using the cycle-free property, we also find such material. We give a production cost for each operating unit producing m as follows. We calculate the sum of the maximum of the *MA* values of the input materials and the c_A cost of the operating unit. (Note that by the definition of m we know that *MA* is known for all input materials.) Then $MA(m)$ will be the minimum of the production costs calculated above for the operating units producing m. After defining *MA* on m we move it from J to I. The procedure ends when J becomes empty which means that *MA* has been calculated for all materials.

Based on this function *MA* we can use two methods to calculate the cumulative cost. In the first case the cumulative cost is the sum of the costs of the input materials plus the direct cost of the operating unit. Using the three direct costs this method yields three cumulative costs and this yields three algorithms. They are denoted by SA (Sum-average), SW (Sum-worst case), SH (Sum-hybrid). A further method to find cumulative costs is to take the sum of the maximal MA cost of the input materials and the direct cost of the operating unit. Again, we can use all of the direct costs thus we obtain three algorithms, which are denoted by MA (Max-average), MW (Max-worst case), MH (Max-hybrid).

We will compare these algorithms in the next section on randomly generated cycle-free inputs. Such cycle-free problems often appear in workflow management. First we describe the test environment and then we analyze the results.

4 Experimental Analysis of the Algorithms

4.1 Description of the Test Cases of the Experimental Analysis

To carry out tests of the algorithms, a problem generator was written. We use similar ideas to the problem generator used in [14] but we changed it to produce cycle free problems. A detailed description of the algorithm which generates the problems is not presented here; only the most important concepts are introduced.

Parameters: The generator has multiple input parameters. We can define the number of operating units to create (*n*), the number of raw materials (*r*) and number of desired products (*p*) in the generated problem. There is an integral variable denoted by w_b that defines how many times it is more probable that an operating unit will reproduce an intermediate material, instead of creating a new one. A further parameter is p_b, which is a probability, related to how likely it is that new operating unit will reproduce intermediate materials that were produced several iterations earlier by other machines. With a low p_b value (e.g. 0.2), newly created operating units will tend to reproduce intermediate materials that are closer to raw materials than to those materials created only some iterations ago. A high p_b value (e.g. 0.8) will introduce redundancy among the production of those intermediate materials that were produced only some iterations ago.

The generator: The generator creates the operating units in an iterative manner. In each iteration, at most one new operating unit and several new materials are created. After the generation of the n'th operating unit, there is a separate step in the algorithm which chooses p materials among those considered intermediate to become product materials. Having chosen the products, the last step is to assign nominal and extra costs to each machine.

The algorithm always keeps a set of materials that are available (W). Initially W only contains the raw materials indexed from 1 to r. An initially empty O set of operating units is maintained throughout the run of the algorithm and at any time it contains the units that are already constructed.

For each new operating unit, two random numbers are chosen in the range [1, 4] using a discrete distribution of (0.15, 0.35, 0.35, 0.15). These two numbers (a, d) represent the number of the inputs and outputs the new operating unit should have. In cases where choosing a number of inputs is impossible, a is truncated to the number of maximum possible inputs a machine might have at that moment.

The core concept of the algorithm is that in each iteration, where there is at most one intermediate material, we choose a limit u that is always greater than r. The operating unit being created, at that iteration, may have inputs only from the range $[1, u-1]$ and outputs from the range $[u, |W|]$. Decisions about whether to create a new material or reproduce an existing intermediate material are made using a dynamic discrete distribution that depends on how many times any intermediate material was chosen for reproduction during the creation of this operating unit, and how many times it was chosen that this operating will produce a new material and a factor that linearly weights these two numbers with a certain bias that is an input parameter of the generator (w_b). Whenever an existing material should be chosen, it is picked from $[u, |W|]$ with a uniform distribution. The u is chosen with binomial distribution over $[r+1, |W|]$ with p_b as its probability parameter.

The fact that any operating unit has a number u that partitions the W set into two parts while inputs can only be chosen from the lower and outputs can only be chosen from the upper partitions guarantees that no solution of the generated problem will contain a cycle.

In the first iteration there are no intermediate materials. This condition forces the algorithm to create an operating unit with all its outputs being new materials.

After a new operating unit is generated, it is inserted to O at the end of the iteration. We note that this set may already contain the same operating unit.

The algorithm first creates $m-1$ operating with the method given above. The method of generating the last operating unit differs from the above, only in that the number of outputs for the last operating unit must guarantee that there will be at least p intermediate materials. This ensures that it is possible to choose p products in the next step.

Having created all of the operating units, uniform integral distributions are used for picking nominal and extra costs for each machine. Random choices of costs are independent from each other.

After the operating units are generated, product materials are chosen from the intermediate ones: p materials are taken from W, whose indices must be in the range $[r+1,|W|]$. The initial probability of material i to be chosen is given by $p_i = (i-r)/S$ where $S =(|W|-r)(|W|-r+1)/2$.

The products are chosen in a non-independent way --- A new material is picked every time with the same distribution introduced above. Whenever such a material is picked that is already considered a product, a new one is picked using the same initial distribution. This method is repeated until p materials are picked. These materials will result in the set of the required products.

We implemented the six heuristic algorithms defined in previously. We used α=0.5 in the average cost in algorithms SA and MA, and also α=0.5 was used in the calculation of the MA function. We generated 1000 operating units and used the values $w_b= 4$ and p_b = 0.2. The number of raw materials was 5, and we generated test cases with the values $p=5, 20, 50, b=1, 5, 25$. We defined the costs of the operating units as the nominal costs picked from the interval [20, 100] uniformly and independently. For the extra cost we considered three different test cases: it was picked from the intervals [0, 50], [20, 100], [100, 500] uniformly and independently. Thus, we defined three possibilities for p, three possibilities for b, and three different distributions for the extra costs. We generated 100 problems from each of these 27 test cases and executed the 6 heuristic algorithms on each test. The results are summarized below.

4.2 The Results of the Experimental Analysis

First consider the test cases where the expected extended cost was smaller than the nominal costs, these cases happens if the uncertainties describe small problems which can be easily handled. The average costs are listed in Table 1. We denoted the best value in each column by bold text and the worst one by bold and italic.

We can see that the order of the efficiencies of the algorithms depends on the test cases. But we can obtain some conclusion.

Table 1
The average results for the extra cost from [0, 50]

	b=1, p=5	b=1 p=20	b=1 p=50	b=5 p=5	b=5 p=20	b=5 p=50	b=25 p=5	b=25 p=20	b=25 p=50
SA	1016	2430	***4167***	1198	2597	4255	1444	***3662***	***5430***
SW	1064	***2542***	**3996**	1341	2627	***4546***	1655	3347	5346
SH	1003	2487	4042	1186	***2781***	4449	**1438**	**3214**	5178
MA	1019	2283	4039	1177	2718	4253	1440	3498	5150
MW	***1074***	2264	4129	***1402***	**2537**	**4209**	***1706***	3220	5323
MH	**997**	**2180**	4150	**1175**	2638	4222	1466	3293	**5082**

Algorithm MH gave the best result 4 times in these test cases and it was never the worst thus we can say that it had the best performance. MW and SH both received the best results two times but MW was the worst 3 times, while SH resulted in the worst solution only in one case thus we can say that SH was better. We also checked the number of best solutions inside the test cases, and we obtained that MH was the best in 239 tests, MW was the best in 146 tests, MA was the best in 189 tests, SH was the best in 142 tests, SW was the best in 76 cases, and SA was the best in 145 cases among the 900 tests. (We note that the sum of the best performance of the algorithms is more than 900. This is not a mistake; if more algorithms achieved the same best result then all of them were considered best. It is likely that in these cases the heuristics found the optimal solution.) The order of the algorithms considering the best results is MH, MA, MW, SA, SH, SW. We also considered the worst cases. Then MH gave 79 times, MW gave 185 times, MA gave 95 times, SH gave 109 times, SW gave 238 times, SA gave 155 times the worst solution. (Here we can observe that the sum is smaller than 900. The reason is that we have not counted here the cases where all of the heuristic found solutions with the same objective value.) The order of the algorithms considering the worst results is MH, MA, SH, SA, MW, SW.

Therefore, our main conclusion that in the tests performed in this block of test cases algorithm MH gave the best results. We cannot clearly order the other heuristics since from different point of views we have different results. But we can observe in general, the maximum evaluation for the indirect costs of the operating units and the hybrid estimation for the direct costs seems to be a better strategy in these sets.

Now, consider the second block where the extra costs have the same distribution as the nominal cost. This block models the situation when the unexpected events are about replacing the operating unit, thus they approximately double the cost of the operating units. The average costs are collected in Table 2, again we denoted the best value in each column by bold, the worst one by bold and italic.

Table 2
The average results for the extra cost from [20, 100]

	b=1, p=5	b=1 p=20	b=1 p=50	b=5 p=5	b=5 p=20	b=5 p=50	b=25 p=5	b=25 p=20	b=25 p=50
SA	1123	2730	4595	1451	2867	4390	**2100**	4539	***6459***
SW	***1165***	***2808***	***4644***	1552	3014	4516	2200	***4823***	6297
SH	1098	2668	4561	1425	**2746**	**4301**	2146	4464	6431
MA	1121	2404	3965	1438	2948	4974	2157	4146	**5857**
MW	1157	2404	3942	***1583***	***3111***	***5042***	***2328***	4510	6524
MH	**1074**	**2308**	**3812**	**1417**	2844	4715	2171	**4133**	6430

Checking the average values we obtain that algorithm MH gave the best result 5 times in these test cases and it was never the worst thus we can again say that it had the best performance. SH received the best result two times and SA and MA both achieved the best average once. Most of the worst results were given by SW and MW both of them achieved a worst result 4 times.

We also checked the number of best solutions inside the test cases, and we obtained that MH was the best in 327 tests, MW was the best in 65 tests, MA was the best in 198 tests, SH was the best in 187 tests, SW was the best in 59 cases, and SA was the best in 86 cases among the 900 tests. The order of the algorithms considering the best results is MH, MA, SH, SA, MW, SW. We considered the worst cases as well. Then MH gave 69 times, MW gave 239 times, MA gave 63 times, SH gave 66 times, SW gave 332 times, SA gave 117 times the worst solution. The order of the algorithms considering the worst results is MA, SH, MH, SA, MW, SW.

Therefore our main conclusion is that in the tests performed in this block of test cases, again, algorithm MH gave the best result. It has slightly more worse cases than MA and SH, but it was much better in the other evaluations. We can also observe that the worst case estimation for the direct costs of the operating units had a very bad performance.

Now consider the last block of the tests, where the extra costs are larger than the nominal cost, this models the situation when the uncertain events cause serious problems which might have larger cost than just replacing the operating unit. The average costs are collected in Table 3, we denoted the best value in each column by bold, the worst one by bold and italic.

Checking the average values we obtain that algorithm MH gave the best result 7 times among the 9 these test cases. It gave the worst result in one case, still we think so that it has the best performance. MW gave the best average in two test cases and it was never the worst thus we can say that it also has a good performance on this block of tests. SW and SA had the worst average results in 5 and 3 cases thus we can say that these were the worse algorithms in this average case.

Table 3
The average results for the extra cost from [100, 500]

	b=1, p=5	b=1 p=20	b=1 p=50	b=5 p=5	b=5 p=20	b=5 p=50	b=25 p=5	b=25 p=20	b=25 p=50
SA	1609	3336	5263	***3086***	5162	6998	6208	***10702***	***14734***
SW	***1740***	***3467***	***5493***	2989	***5202***	***7310***	6113	10083	14050
SH	1596	3274	5235	3003	5038	7117	6198	10510	14333
MA	1715	3225	4951	3004	4953	6837	6220	10653	13542
MW	1746	3257	5137	3020	5066	7067	**6051**	**10024**	13312
MH	**1579**	**2984**	**4648**	**2887**	**4706**	**6785**	***6284***	10600	**13180**

We checked the number of best solutions inside the test cases, and we obtained that MH was the best in 424 tests, MW was the best in 144 tests, MA was the best in 107 tests, SH was the best in 103 tests, SW was the best in 63 cases, and SA was the best in 68 cases among the 900 tests. The order of the algorithms considering the best results is MH, MW, MA, SH, SA, SW. We also considered the worst cases as well. Then MH gave 47 times, MW gave 126 times, MA gave 106 times, SH gave 120 times, SW gave 256 times, SA gave 255 times the worst solution. The order of the algorithms considering the worst results is MH, MA, SH, MW, SA, SW.

Conclusions

Therefore, our main conclusion, that in the tests performed in this block of test cases, that the algorithm MH gave the best result. We can also observe that the sum estimations for the indirect costs of the operating units in general had worse performance.

Summarizing the experimental analysis we can observe that the performance of the algorithms strongly depends on the test cases, but we can draw some conclusion here. Algorithm MH had clearly the best result in each block of test cases, thus we can state that the experiments show that MH is the most efficient algorithm among the algorithms studied in this work. We could note that the difference between it and the other algorithms is the most significant when the extra costs are large. On the other hand SW was the worst in most cases. In general we can conclude that the maximum estimations perform in a better way for the indirect costs than the sum ones. And we can also conclude that the worst case estimation on the direct cost of the operating units has poorer performance than the other methods.

Acknowledgement

This work was partially supported by the European Union and the European Social Fund through project Telemedicina (Grant no.: TÁMOP-4.2.2.A-11/1/KONV-2012-0073).

References

[1] Bárány, M.; Bertók, B.; Kovács, Z.; Friedler, F.; Fan, L. T.: Solving Vehicle Assignment Problems by Process-Network Synthesis to Minimize Cost and Environmental Impact of Transportation, Clean Technologies and Environmental Policy, 13(4), 637-642, 2011

[2] Bertók, B.; Kalauz, K.; Süle, Z.; Friedler, F.: Combinatorial Algorithm for Synthesizing Redundant Structures to Increase Reliability of Supply Chains: Application to Biodisel Supply, Industrial & Engineering Chemistry Research, 52(1), 181-186, 2013

[3] Bertsimas, D.; Brown, D.; Caramanis, C.: Theory and Applications of Robust Optimization, SIAM Review, 53, 464-501, 2011

[4] Bertsimas, D.; Sim, M.: Robust Discrete Optimization and Network Flows, Mathematical Programming Series B, 98, 49-71, 2003

[5] Blázsik, Z.; Holló, Cs.; Imreh, Cs.; Kovács, Z.: Heuristics for the PNS problem, Optimization Theory, Mátraháza 1999, Applied Optimization 59 eds. F. Gianessi, P. Pardalos, T. Rapcsák, Kluwer Academic Publishers, Dordrecht, Boston, London, 1-18, 2001

[6] Blázsik, Z.; Imreh, B.: A Note on Connection between PNS and Set Covering Problems, Acta Cybernetica, 12, 309-312, 1996

[7] Blázsik, Z.; Keserű K.; Kovács, Z.: Heuristics for Simplified Process Network Synthesis Problems with a Blossom-Type Algorithm for the Edge Covering Problem, Optimization Theory, Mátraháza 1999, Applied Optimization 59 eds. F. Gianessi, P. Pardalos, T. Rapcsák, Kluwer Academic Publishers, Dordrecht, Boston, London, 19-31, 2001

[8] Fan, L.T.; Kim, Y.; Yun, C.; Park, S. B.; Park, S.; Bertok, B.; Friedler, F.: Design of Optimal and Near-Optimal Enterprise-Wide Supply Networks for Multiple Products in the Process Industry, Ind. Eng. Chem. Res., 48, 2003-2008, 2009

[9] Friedler, F.; Fan, L. T.; Imreh, B.: Process Network Synthesis: Problem Definition, Networks, 28, 119-124, 1998

[10] Friedler, F.; Tarján, K.; Huang, Y. W.; Fan, L.T.: Graph-Theoretic Approach to Process Synthesis: Axioms and Theorems, Chem. Eng. Sci., 47(8), 1973-1988, 1992

[11] Garcia-Ojeda, J. C.; Bertók, B.; Friedler, F.: Planning Evacuation Routes with the P-Graph Framework, Chemical Engineering Transactions, 29, 1531-1536, 2012

[12] Garcia-Ojeda, J. C.; Bertók, B.; Friedler, F.; Fan, L. T.: Building-Evacuation-Route Planning via Time-expanded Process-Network Synthesis, Fire Safety Journal, 61, 338-347, 2013

[13] Imreh, B.; Friedler, F.; Fan, L. T.: An Algorithm for Improving the Bounding Procedure in Solving Process Network Synthesis by a Branch-and-Bound Method Developments in Global Optimization, editors: I. M. Bonze, T. Csendes, R. Horst, P. M. Pardalos, Kluwer Academic Publisher, Dordrecht, Boston, London, 301-348, 1996

[14] Imreh, B.; Magyar, G.: Empirical Analysis of Some Procedures for Solving Process Network Synthesis Problem, Journal of Computing and Information Technology, 6, 372-382, 1998

[15] Süle, Z.; Bertók, B.; Friedler, F.; Fan, L. T.: Optimal Design of Supply Chains by P-Graph Framework Under Uncertainties, Chemical Engineering Transactions, 25, 453-458, 2011

[16] Tick, J.: Fuzzy Extension to P-Graph-based Workflow Models, Proceedings of the 7th IEEE International Conference on Computational Cybernetics, ICCC 2009, 109-112, 2009

[17] Tick, J.: P-Graph-based Workflow Modeling, Acta Polytechnica Hungarica (ISSN: 1785-8860) 4: (1), 75-88, 2007

[18] Tick, J.: Workflow Modeling Based on Process Graph, Proceedings of the 5th Slovakian-Hungarian Joint Symposium on Applied Machine Intelligence and Informatics (SAMI 2007) Poprad, Slovakia, 2007.01.25-26, 419-426, 2007

[19] Tick, J.; Imreh Cs.; Kovács Z.: Business Process Modeling and the robust PNS problem, Acta Polytechnica Hungarica (ISSN: 1785-8860) 10: (6) 193-204, 2013

[20] Tick, J.; Kovács, Z.: P-Graph-based Workflow Synthesis, Proceedings of the 12th International Conference on Intelligent Engineering Systems (INES 2008) Miami, USA, 2008.02.25-29, 249-253, 2008

[21] Tick, J.: Fuzzy Control Systems Based on Parametric T-Norm Function, Proceedings of the 4th International Symposium on Applied Computational Intelligence and Informatics (SACI 2007) Timisoara, Romania, 2007.05.17-18, 215-218, 2007

[22] Vance, L.; Cabezas, H.; Heckl, I.; Bertók, B.; Friedler, F.: Synthesis of Sustainable Energy Supply Chain by the P-Graph Framework, Industrial & Engineering Chemistry Research, 52(1), 266-274, 2013