Configurations of Series-Parallel Networks with Maximum Reliability

Walter Gutjahr
Georg Ch. Pflug
Andrzej Ruszczyński

WP-93-60
October 1993

Configurations of Series-Parallel Networks with Maximum Reliability

Walter Gutjahr
Georg Ch. Pflug
Andrzej Ruszczyński

WP-93-60
October 1993

IIASA International Institute for Applied Systems Analysis ◻ A-2361 Laxenburg ◻ Austria
Telephone: +43 2236 715210 ◻ Telex: 079 137 iiasa a ◻ Telefax: +43 2236 71313

Configurations of Series-Parallel Networks with Maximum Reliability

Walter Gutjahr [*], *Georg Ch. Pflug* [*], *Andrzej Ruszczyński* [**]

[*] Institute of Statistics, Operations Research and Computer Science, University of Vienna, Austria

[**] International Institute for Applied Systems Analysis, Laxenburg, Austria

Keywords:
open failure, shorted failure, series-parallel networks, reliability.

Reader aids:
Purpose: Provide an algorithm for designing optimal network configurations.
Special math needed: Elementary probability, calculus.
Special math needed to use results: Elementary arithmetics.
Results useful to: System reliability analysts.

Summary:
The optimal design problem for networks with 3-state components is the following: select from a given class of networks with n components, each of which can be operative or experience an open-mode or a shorted-mode failure state, the network with maximum reliability. We present an algorithm for solving this problem in the case of 2-stage series-parallel networks, i. e., networks consisting of a number of series configurations linked in parallel or vice versa. For practically relevant network sizes (up to 100 components), the algorithm is fast.

1 Introduction

Networks composed of 3-state devices have been intensively investigated in the literature on reliability. A 3-state device is one which can, besides its normal operative state, assume two different failure states: an *open-mode* and a *shorted-mode* failure state. Corresponding to these failure possibilities, a network (with specified source and sink) whose components are 3-state devices can either be operative, or experience an "open failure" or a "shorted failure".

- An *open failure* occurs if every path through the system (from source to sink) contains at least one component in the open-mode failure state.

- A *shorted failure* occurs it there exists a path through the system (from source to sink) consisting only of components in the shorted-mode failure state.

The *optimal design problem* is the following: select from a given class of networks with n components one which has maximal reliability, i.e., minimal probability that the system fails in either the open mode or the shorted mode.

Usually, it is assumed that both

$$q = \text{probability of an open-mode failure}$$

and

$$s = \text{probability of a shorted-mode failure}$$

are the same for each component, and that failures occur independently from each other. In every case $q + s < 1$.

For a more detailed description of the problem, examples of applications and further references, see [5]. This type of problems was also thoroughly investigated in [7] and in [8].

As to the class under consideration, Page and Perry [5] investigated *series-parallel networks*, i.e., networks consisting either of a single component or of series-parallel subnetworks connected to each other in series or parallel (cf. Fig. 1.). Because of the computational complexity of the problem, only networks up to nine components were treated by exact optimization.

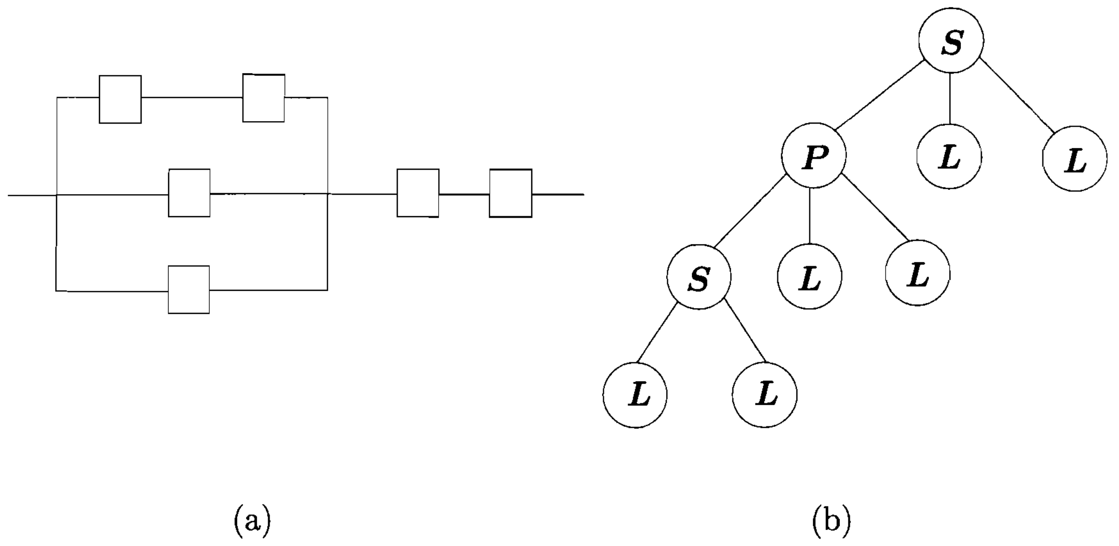

Fig. 1. A series-parallel network. (a): the usual graph representation; (b): tree representation. S means series, P means parallel and the leaves L are single components.

In the present article, we will restrict ourselves to 2-stage series-parallel networks. It will be shown that for this class even 100 and more components can be dealt with by means of an exact optimization algorithm.

A *2-stage series-parallel network* is a series-parallel network with the property that, in its tree representation, each leaf has depth 2 (cf. Fig. 2.). Notice that we do not only allow binary trees, as Page and Perry [5] did, but general (rooted) trees. There are two types of such networks:

(1) *PS-networks* (cf. Fig. 2 (a)). They consist of a number of series configurations of components, linked in parallel.

(2) *SP-networks* (cf. Fig. 2 (b)). They consist of a number of parallel configurations of components, linked in series.

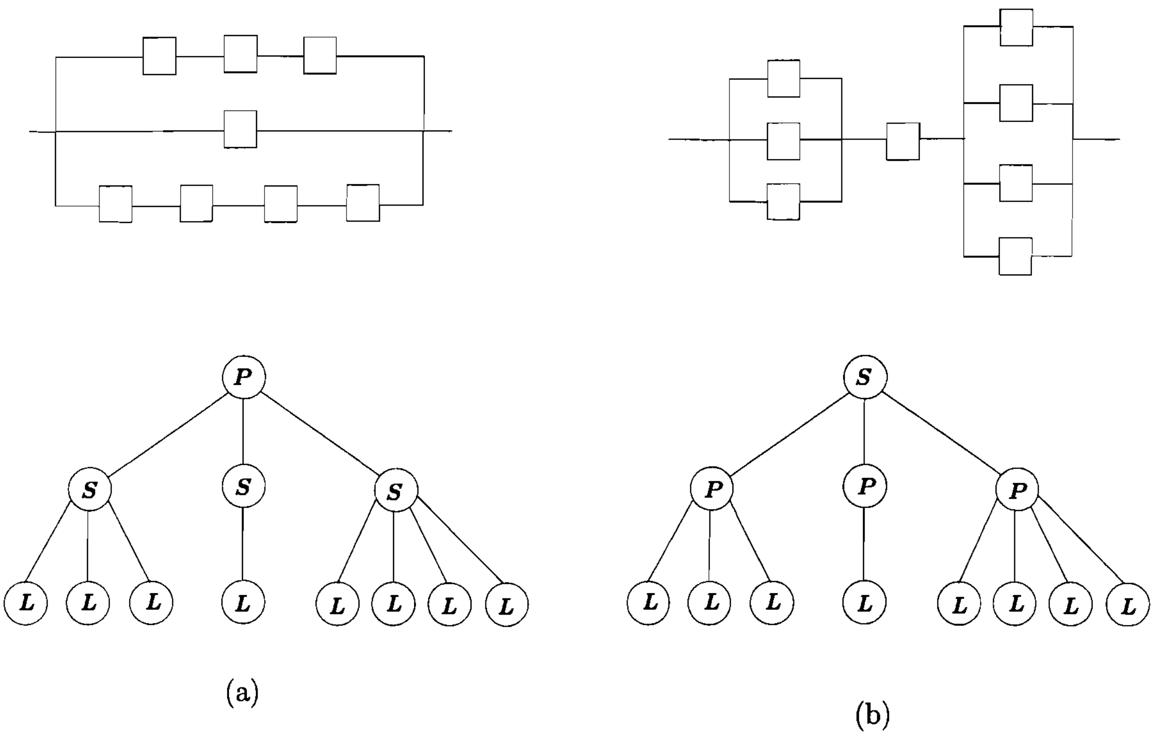

Fig. 2. Two examples for 2-stage series-parallel networks.
(a): a PS-network; (b): an SP-network.

Special cases of 2-step series-parallel networks are the so-called *arrays*, where the number of components in each series configuration (resp. in each parallel configuration) is constant. The reliability of arrays with 3-state components was investigated in [3].

In the sequel, we will restrict ourselves to PS-networks. The results for SP-networks follow immediately by exchanging symbols S and P in the tree representation and by exchanging open-mode and shorted-mode failure probabilities in the equations.

Formulas for the reliability of PS-networks may easily be derived, using elementary probability theory. For a PS-network, let x_i denote the number of components in the ith series configuration. By m we denote the number of series configurations. (In the example of Fig. 2 (a), $x_1 = 3, x_2 = 1, x_3 = 4$ and $m = 3$.) Let $\boldsymbol{x} = (x_1, \ldots, x_m)$. It should be noted that a PS-network is completely determined by $\boldsymbol{x}$. Then with q, s as above, the total failure probability (i. e. the sum of the open and the shorted failure probability) is given by

$$F(\boldsymbol{x}) = \prod_{i=1}^{m}(1 - (1 - q)^{x_i}) + 1 - \prod_{i=1}^{m}(1 - s^{x_i}) \qquad (1)$$

(cf. [7], p. 102).

The optimal design problem may now be formulated as follows: For given n, q and s, find an integer $m > 0$ and a vector $\boldsymbol{x} = (x_1, \ldots, x_m)$ of integers $x_i > 0$, such that

$$\left\| \begin{array}{ccc} F(\boldsymbol{x}) & \to & \min \\ \sum\limits_{i=1}^{m} x_i & = & n. \end{array} \right. \qquad (2)$$

2 Optimization by complete enumeration

Page's and Perry's [5] solution strategy for the optimal design problem is *complete enumeration*: they compute all series-parallel networks of a given size (i.e., number of components) that are nonequivalent with respect to their reliability, and select the best one. Clearly, complete enumeration is also possible for the special case of PS- (or SP-) networks. We shall describe such an enumerative approach which is efficient for small n.

Let us start with the observation that interchanging the elements $x_1, \ldots, x_m$ in a vector $\boldsymbol{x}$ describing a PS-network leads to an equivalent network (a network with identical reliability for arbitrary q and s as the original one). For example, the PS-network described by $\boldsymbol{x} = (3, 4, 1)$ has always the same reliability as the network of Fig. 2 (a), described by $\boldsymbol{x} = (3, 1, 4)$. So it suffices to check all *partitions* of the given integer n, i.e., all representations of n as unordered sums of positive integers (see, e.g., [6], pp. 107-124). In the case $n = 5$, for example, there are seven partitions, namely 5, 4+1, 3+2, 3+1+1, 2+2+1, 2+1+1+1 and 1+1+1+1. We shall also use the notation (4, 1) instead of $4 + 1$, etc. Each partition corresponds to a PS-network, the failure probability of which may be determined from (1). The network with minimal value $F(\boldsymbol{x})$ yields the solution.

In order to minimize space and time when running through the partitions, it is useful to apply a partition generating algorithm of "next"-type, i.e., an algorithm which takes a partition $\boldsymbol{x}$ as input and computes the *next* partition $\boldsymbol{x}'$ in a pre-specified order for all partitions of fixed integer n as output. Such an algorithm was described by Nijenhuis and Wilf ([4], pp. 63-69). They chose antilexicographic order, i. e., the first partition is (n), the last partition is $(1, \ldots, 1)$. Since this algorithm will be used again in section 5, we cite it here (without much explanation) for the ease of the reader:

A partition $\boldsymbol{x}$ is represented by two arrays:

- $r[1] > \ldots > r[d]$ are the distinct integers occuring in $\boldsymbol{x}$,
- $m[1], \ldots, m[d]$ are their respective multiplicities.

First partition:
$r[1] := n$; $m[1] := 1$; $d := 1$.

Switch to the next partition:

```
procedure next-partition ($\boldsymbol{x}$)
begin
if $r[d] = 1$ then begin $\sigma := m[d] + 1$; $d := d - 1$ end
        else $\sigma := 1$;
$f := r[d] - 1$;
if $m[d] > 1$ then begin $m[d] := m[d] - 1; d := d + 1$ end;
$r[d] := f$;
$m[d] := \lfloor \sigma / f \rfloor + 1$;
$s := \sigma \pmod f$;
if $s > 0$ then begin $d := d + 1; r[d] := s; m[d] := 1$ end;
if $m[d] = n$ then final exit (*  last partition reached  *);
end.
```

During the enumeration process only the actual partition $\boldsymbol{x}$ and the best solution $\boldsymbol{x}^*$ found up to that time has to be stored.

For $n = 50$, an optimization run required about 4 minutes on a PC 486.

Even by this approach, however, sizes of (say) $n = 100$ are beyond the limits of tractability. According to a formula by De Bruijn [2], the number $p(n)$ of partitions of n is asymptotically

$$p(n) \sim \frac{1}{4\sqrt{3} \cdot n} \exp\left(\pi\sqrt{2n/3}\right).$$

For $n = 100$, this already yields $p(100) \approx 2 \cdot 10^8$ partitions to be examined. (Compare with $p(50) = 204226$.) Thus, the solution space of the optimization problem has to be restricted in order to obtain a more efficient algorithm for large n.

3 The continuous analogue

In order to get more information about the problem (2), we consider its *continuous analogue*, obtained by dropping the condition of integrality for the numbers x_i. Now the values x_i may be arbitrary real numbers > 0. With

$$r := 1 - q$$

this yields the problem

$$\left\|\begin{array}{l}\prod_{i=1}^m(1-r^{x_i})+1-\prod_{i=1}^m(1-s^{x_i})\to\min\\ \sum_{i=1}^m x_i=n\\ x_i>0\quad(i=1,\dots,m).\end{array}\right. \tag{3}$$

Therein, m is also a variable to be optimized. At first, however, let us fix m. Since $q+s<1$, one has $s<r$.
Let now $y_i = x_i/n$ and

$$r^n=\exp(-\rho),\ s^n=\exp(-\sigma) \tag{4}$$

such that $\sigma>\rho$.
Then (3) may be rewritten as

$$\left\|\begin{array}{l}\prod_{i=1}^m(1-\exp(-\rho y_i))+1-\prod_{i=1}^m(1-\exp(-\sigma y_i))\to\min\\ \sum_{i=1}^m y_i=1\\ y_i>0\qquad(i=1,\dots m).\end{array}\right. \tag{5}$$

The set of feasible solutions is now the (open) m-dimensional standard simplex.

For abbreviation, let us define

$$h(\rho,\boldsymbol{y})=\prod_{i=1}^m(1-\exp(-\rho y_i)). \tag{6}$$

Because of the differentiability of the objective function $h(\rho,\boldsymbol{y})+1-h(\sigma,\boldsymbol{y})$ in (5), a necessary condition for a point $\boldsymbol{y}$ in the set of feasible solutions (the open simplex) to be an optimal solution is *stationarity*, i.e.,

$$\nabla h(\rho,\boldsymbol{y})-\nabla h(\sigma,\boldsymbol{y})=\mu\mathbf{1} \tag{7}$$

with $\mathbf{1}=(1,\dots,1)^t$ and an appropriate Lagrange multiplier $\mu\in R$.
From (7), with

$$g_\rho(y)=(\exp(\rho y)-1)^{-1} \tag{8}$$

one obtains

$$\rho\, h(\rho,\boldsymbol{y})\, g_\rho(y_i)-\sigma\, h(\sigma,\boldsymbol{y})\, g_\sigma(y_i)=\mu\qquad(i=1,\dots,m).$$

Thus

$$g_\rho(y_i)=c_1 g_\sigma(y_i)+c_2\qquad(i=1,\dots,m) \tag{9}$$

with constants c_1,c_2 independent of i.

It is possible to show the following property:

Lemma 3.1. *The equation*

$$g_\rho(y) = c_1 g_\sigma(y) + c_2 \tag{10}$$

has at most two solutions in $y > 0$.

Proof. With $u = \exp(\sigma y)\,(u > 1)$ and $\alpha = \rho/\sigma$ $(0 < \alpha < 1)$, equation (10) reads

$$\frac{u-1}{u^\alpha - 1} = c_1 + c_2(u-1). \tag{11}$$

The right hand side of (11) is linear. We show that $\varphi(u) = (u-1)/(u^\alpha - 1)$ is strictly concave for $u > 1$; hence the assertion follows, since a strictly concave function cannot have more than two intersection points with a linear one.
One computes

$$\varphi''(u) = f_1(u)f_2(u)$$

with

$$f_1(u) = \alpha u^{\alpha-2}(u^\alpha - 1)^{-3} > 0,$$

and

$$f_2(u) = (\alpha-1)u^{\alpha+1} - (\alpha+1)u^\alpha + (\alpha+1)u - (\alpha-1).$$

Furthermore, it is easily verified that $f_2''(u) < 0$ for all $u > 1$. Because of $f_2(1) = f_2'(1) = 0$, also $f_2(u) < 0$ for all $u > 1$. Therefore $\varphi''(u) < 0$ $(u > 1)$. □

Lemma 3.1 allows a drastic reduction of the set of possible optimal solutions of (5). From (9), we immediately obtain:

Proposition 3.1. *To each optimal solution* $\boldsymbol{y}^* = (y_1^*, \dots y_n^*)$ *of (5) there are two values* a, b, *such that* $y_i^* = a$ *or* $y_i^* = b$ $(1 \le i \le n)$. *In other words: the components* y_i^* *of an optimal solution cannot assume more than two different values.*

Let a, b be the different values assumed by the variables y_i^* $(a \ge b)$. Furthermore, let a and b occur k times and $m-k$ times in $\boldsymbol{y}^*$, respectively; without loss of generality we may assume $y_1^* = \dots = y_k^* = a$, $y_{k+1}^* = \dots = y_m^* = b$. Then $ka + (m-k)b = 1$. From $a \ge b$ follows $b \le 1/m$. Hence with $\lambda = bm$, an optimal solution $\boldsymbol{y}^*$ of (5) may be written as

$$y_i^* = \begin{cases} \frac{\lambda}{m} + \frac{1-\lambda}{k}, & 1 \le i \le k, \\ \frac{\lambda}{m}, & k+1 \le i \le m, \end{cases} \tag{12}$$

where $0 \le \lambda \le 1$.

The representation (12) of the solution still contains three parameters m, k, λ to be optimized. In the next section it will be shown how this can be done numerically for given q, s. Clearly, it would be desirable to obtain a closed formula for the optimal solution of (5). Because of the highly nonlinear (and even multi-extremal) character of the optimization problem, this seems not achievable. Our numerical observations led to the following conjecture, but we were not able to prove it:

Conjecture. In an optimal solution $\boldsymbol{y}^*$ of (5) with an optimal value $m = m^*$, all components y_i^* are equal. (I.e., $\lambda = 1$ and k arbitrary in (12)).

4 An approximation algorithm

An approximate solution of (2) can be found by the following algorithm.

For all $m = 1, \dots, n$:

(a) Compute an approximate solution for the continuous problem (5) with fixed m by optimizing numerically the parameters k and λ in (12). This yields variables $\tilde{x}_i^{(m)} = ny_i^{*(m)} > 0$ $(i = 1, \dots, m)$ with $\sum \tilde{x}_i^{(m)} = n$.

(b) Round the variables $\tilde{x}_i^{(m)}$ to integers $x_i^{(m)}$, in such a way that the condition $\sum x_i^{(m)} = n$ remains satisfied.

(c) Compute by (1) the objective function $F(\boldsymbol{x}^{(m)})$ for $\boldsymbol{x}^{(m)} = (x_1^{(m)}, \dots, x_m^{(m)})$.

The final solution is $\boldsymbol{x}^* = \boldsymbol{x}^{(m)}$ for that m which minimizes $F(\boldsymbol{x}^{(m)})$ $(m = 1, \dots, n)$.

In the theory of integer programming it is well known that roundings of an optimal solution of a continuous optimization problem need *not* produce an optimal solution of the corresponding *integer* optimization problem ([1], p. 715). So even if, in step (a) above, we find the exact solution $\tilde{\boldsymbol{x}}^{(m)}$, there is no guarantee that the final solution is optimal for the original problem (2). Thus the algorithm above is only a heuristics, albeit a rather good one, as will be seen in the next section.

Step (a) and (b) of the algorithm will now be specified in more detail:

(a) In principle all possible values $k = 1, \dots, m$ have to be checked. (As far as we observed, already $k = 1$ leads to the optimum in each case, but we have no proof for that). For fixed k (and m), the optimal value λ can be found as follows: With $f(\rho, \lambda) = \left[1 - \exp\left(-\frac{\rho\lambda}{m} - \frac{\rho(1-\lambda)}{k}\right)\right]^k \left[1 - \exp\left(-\frac{\rho\lambda}{m}\right)\right]^{m-k} = h(\rho, \boldsymbol{y}^*)$, where $\boldsymbol{y}^*$ is defined by (12), the problem

$$\left\| \begin{array}{l} \tilde{F}(\lambda) = f(\rho, \lambda) + 1 - f(\sigma, \lambda) \to \min \\ 0 \le \lambda \le 1 \end{array} \right. \qquad (13)$$

has to be solved. This can be done numerically by line search. For example, take $t + 1$ equidistant search points $\lambda = 0,\ 1/t,\ 2/t,\ \dots,\ (t-1),\ t,\ 1$. It is not difficult to see that the function $f(\rho, .)$ is Lipschitz-continuous with Lipschitz constant ρ; as a consequence, $\tilde{F}(.)$ is Lipschitz-continuous with Lipschitz constant

$$\rho + \sigma = -\log(rs) \cdot n = Kn.$$

Now choose, for given $\epsilon > 0$, the integer t as $t = \lceil Kn/\epsilon \rceil$. Then the objective function value $\tilde{F}(\bar{\lambda})$ obtained from the optimal search point $\bar{\lambda}$ differs from the minimal objective function value $\tilde{F}(\lambda^*)$ only by ϵ or less, so (13) can be solved to any given accuracy ϵ.

(b) In the case where all components $\tilde{x}_i^{(m)}$ lie in the same interval $[j, j+1]$, there is a unique way to round the real numbers $\tilde{x}_i^{(m)}$ to integers $x_i^{(m)}$, such that $\sum x_i^{(m)} = n$ and $x_i^{(m)} \geq x_{i+1}^{(m)}$ $(i = 1, \dots, m-1)$.
Otherwise, there may be different possible ways to do the rounding. Since, according to our experience gathered so far, an optimal value m always leads to a solution $\tilde{\boldsymbol{x}}^{(m)}$ with $\tilde{x}_1^{(m)} = \dots = \tilde{x}_m^{(m)}$, it is not important how to treat the latter case: any way of rounding may be chosen.

Clearly, application of a faster search method than equidistant search in step (a) of the algorithm (for example binary search or Golden Section search) would be desirable. Unfortunately, however, $\tilde{F}(\lambda)$ is not unimodular in all cases; hence local search methods are insufficient. To see this, consider e. g. the case $m = 5$, $k = 1$, $\rho = 10$ and $\sigma = 16$. Then, as Fig. 3 shows, there are two local minima.

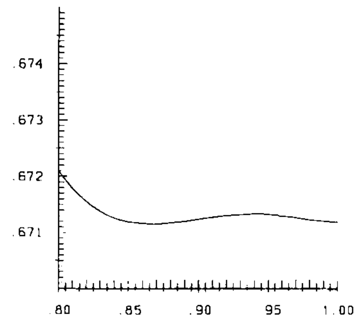

Fig. 3. $\tilde{F}(\lambda)$ for $m = 5$, $k = 1$, $\rho = 10$ and $\sigma = 16$ in the interval [0.8,1].

Since the determination of $\overline{\lambda}$, as described above, requires $O(n)$ time, the time required by the whole procedure is of order $O(n^3)$. For $n = 100$, the computation took about one minute on a PC 486.

5 Fast exact optimization by solution inprovement

As already mentioned, the algorithm described in section 4 may possibly stop with a suboptimal solution. The aim of the present section is to develop a strategy of solution improvement: If a "good" (but possibly not optimal) feasible solution $\boldsymbol{x}$ has been found, we use information delivered by $w = F(\boldsymbol{x})$ in order to find an optimal solution $\boldsymbol{x}^*$.

For this purpose, two estimations will be considered:

Estimation 1:
Let $\boldsymbol{x}^* = (x_1^*, \dots, x_m^*)$ be optimal, and let $w = F(\boldsymbol{x})$ be the value of the objective function (1) for an arbitray feasible solution $\boldsymbol{x}$. Then for each $i = 1, \dots, m$:

$$s^{x_i^*} \leq F(\boldsymbol{x}^*) \leq w, \tag{14}$$

since the probability of a shorted failure caused by the ith series configuration is a lower bound for the total failure probability.
By taking logarithms and observing that x_i^* is an integer, we obtain

$$x_i^* \geq \left\lceil \frac{\log w}{\log s} \right\rceil = k_o \quad (i = 1, \ldots, m), \tag{15}$$

so k_o is a lower bound for the number of components in each series configuration of an optimal network.

Estimation 2:
With $x^* = (x_1^*, \ldots, x_m^*)$ and k_o as above, consider a PS-network $\boldsymbol{x}^o$ consisting of m series configurations, each with *exactly* k_o components (while the optimal network $\boldsymbol{x}^*$ consists of m series configurations with *at least* k_o components). The open failure probability of $\boldsymbol{x}^o$ is $(1 - r^{k_o})^m$, and obviously this is a lower bound for the open failure probability of $\boldsymbol{x}^*$. Thus with w as above,

$$(1 - r^{k_o})^m \leq F(\boldsymbol{x}^*) \leq w. \tag{16}$$

Again by taking logarithms, one finds

$$m \geq \left\lceil \frac{\log w}{\log(1 - r^{k_o})} \right\rceil = m_o. \tag{17}$$

Thus we obtain lower bounds k_o and m_o for the "width" and the "height" of an optimal network, respectively. Clearly, these bounds are the better, the nearer $w = F(\boldsymbol{x})$ gets to the optimal objective function value $F(\boldsymbol{x}^*)$. It is therefore important to start with a sufficiently good feasible solution $\boldsymbol{x}$.

By using the information on minimal width and height, the position of $k_o m_o$ components in an optimal network is already determined. The remaining $n - k_o m_o$ components may be arranged optimally by complete enumeration of partitions:

Example 5.1. Let $n = 20$ and $q = s = 0.1$. We start with the feasible solution $\boldsymbol{x} = (4, 4, 3, 3, 3, 3)$ produced by the algorithm in section 4. One computes $w = F(\boldsymbol{x}) = 0.0048311$, and hence $k_o = 3, m_o = 5$. So already 15 of the 20 components can be arranged (cf. Fig. 4).

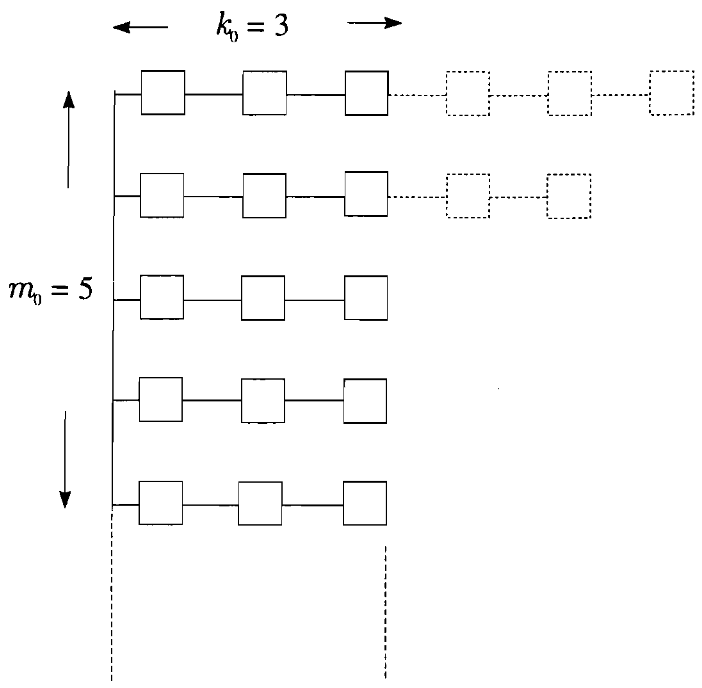

Fig. 4. Construction of an optimal network in Example 5.1, using the estimates k_0 and m_0.

The remaining 5 components have to be arranged by complete enumeration of partitions:

- For $m = 5$, each partition of $n - k_o m = 5$ into $m = 5$ or less parts has to be added to the partition (3, 3, 3, 3, 3), where addition is performed componentwise (if necessary, fill up with zeros to the right hand side). This yields the partitions
  (8, 3, 3, 3, 3),
  (7, 4, 3, 3, 3),
  (6, 5, 3, 3, 3) (shown in Fig. 4.),
  (6, 4, 4, 3, 3),
  (5, 5, 4, 3, 3),
  (5, 4, 4, 4, 3),
  (4, 4, 4, 4, 4).
  The failure probabilities of the corresponding networks have to be computed.

- For $m = 6$, each partition of $n - k_o m = 2$ into $m = 6$ or less parts has to be added to the partition (3, 3, 3, 3, 3, 3). This yields the partitions
  (5, 3, 3, 3, 3, 3),
  (4, 4, 3, 3, 3, 3),
  Again, failure probabilities have to be computed.

- $m = 7$ is not possibly anymore, since $3 \cdot 7 = 21 > 20$.

Among the 9 enumerated partitions, (4, 4, 3, 3, 3, 3) leads to minimal failure probability, so the initial solution is already optimal.

◇

We give now a general description of the optimization algorithm:

(1) Compute an approximate solution $\boldsymbol{x}$ and $w = F(\boldsymbol{x})$ by the algorithm of section 4.

(2) Compute the lower bounds k_o and m_o from (15) and (17).

(3) Find the optimum by complete enumeration of the partitions satisfying the bounds determined by k_o and m_o.

In more detail, step (3) consists of the following procedure:

```
for $m := m_o$ to $\lfloor n/k_o \rfloor$ do
begin for all partitions $\boldsymbol{u}$ of $n - k_o m$
          if length $(\boldsymbol{u}) \leq m$
          then begin add $\boldsymbol{u}$ to the partition
               $(k_o, \ldots, k_o)$ ($m$ parts), yielding partition $\boldsymbol{z}$;
               compute $F(\boldsymbol{z})$ and store $\boldsymbol{z}$ if it is better than the
               best previously found partition
               end
end.
```

The partitions of $n - k_o m$ can be determined successively by the procedure next-partition described in section 2.

Clearly, also the algorithm above may lead to high execution costs, if the number $n - k_o m_o$ of remaining components gets large. It turns out, however, that for "reasonable" sizes of n, the integer $n - k_o m_o$ remains comparatively small, especially if q and s are small (which is a realistic assumption). In the case $n = 100, q = 0.1$ and $s = 0.001$, e.g., only $n - k_o m_o = 8$ of the 100 components have to be arranged by complete enumeration.

For $n = 100, q = 0.1$ and $s = 0.1$, the number of components to be arranged by complete enumeration is 40. Fig. 5. shows a comparison of execution times required for $q = 0.1$ and $s = 0.1$. The reader should notice that even in the case $n = 100$ the larger part of the execution time is consumed by step (1) of the algorithm, i.e., the computation of an approximate solution according to the algorithm of section 4. This algorithm, however, has only the polynomial worst case behavior of $O(n^3)$.

n	I	II	III
20	< 1	2	2
30	6	4	4
40	36	6	6
50	229	11	11
60	1320	19	21
70	> 10000	26	39
80		40	43
90		56	65
100		78	99

Fig. 5. Execution times (in seconds) for $q = 0.1$ and $s = 0.1$.
I: Complete enumeration (section 2); II: approximation algorithm (section 4);
III: approximation plus solution improvement (section 5).

Interestingly enough, we *never* encountered a case where the solution determined by the algorithm of section 4 turned out to be suboptimal. Thus, the above-described solution improvement procedure only served as a *verification* of optimality. For practical applications where near-to-optimal solutions are also acceptable, this verification may be dropped at all. Then even very large network sizes can be treated.

Fig. 6. shows optimal values of m for $n = 20$ and different failure probabilities q, s. If the optimal m for some n, q, s is known, an optimal or near-to-optimal PS-network can be construced in the following way (cf. the conjecture in section 3 and the observation above):

- Build up m series configurations, each with $\lfloor n/m \rfloor$ components.
- To $n - \lfloor n/m \rfloor \cdot m$ of the m series configurations, add an additional component.

These solutions are slight modifications of *arrays* (cf. section 1).

Page and Perry [5] observed that series-parallel networks that are optimal (among series-parallel networks) for any q, s, also occur as optimal networks in an arbitrary small neighborhood of $(q, s) = (0, 0)$. In our own experiments, we observed an analogous property for the class of PS-networks: No value m "dies out" as a possible optimal network height as (q, s) tends to the origin. In view of this, there is little hope that the case of very small q and s might lead, via asymptotic considerations, to closed-form solutions; application of an optimization algorithm seems to be indispensible also for this case.

q \ s	0.01	.	.	.	.	.	.	.	.	0.10
0.01	5	5	5	4	4	4	4	4	4	4
0.02	5	5	5	5	5	5	4	4	4	4
.	6	5	5	5	5	5	5	5	5	4
.	6	6	5	5	5	5	5	5	5	5
.	6	6	6	5	5	5	5	5	5	5
.	6	6	6	6	5	5	5	5	5	5
.	6	6	6	6	6	6	5	5	5	5
.	7	6	6	6	6	6	6	5	5	5
.	7	6	6	6	6	6	6	6	5	5
0.10	7	6	6	6	6	6	6	6	6	6

q \ s	0.001	.	.	.	.	.	.	.	.	0.010
0.001	5	5	5	4	4	4	4	4	4	4
0.002	5	5	5	5	5	5	5	4	4	4
.	5	5	5	5	5	5	5	5	5	4
.	5	5	5	5	5	5	5	5	5	5
.	5	5	5	5	5	5	5	5	5	5
.	6	5	5	5	5	5	5	5	5	5
.	6	5	5	5	5	5	5	5	5	5
.	6	5	5	5	5	5	5	5	5	5
.	6	6	5	5	5	5	5	5	5	5
0.010	6	6	5	5	5	5	5	5	5	5

Fig. 6. Optimal network height m for $n = 20$ and different values q, s.

6 Conclusions

We have presented a reliability optimization algorithm for 2-stage (PS- or SP-) series-parallel networks with a given number n of components that may experience open-mode or shorted-mode failures. The algorithm consists of two parts: a heuristics, based on the continuous analogue of the optimization problem, and a solution-improving mechanism finding with certainty an optimal solution. In all cases observed by us, already the heuristics yielded an optimal solution which was then confirmed by the solution-improving mechanism. The described algorithm is fast enough to deal with all network sizes of practical interest.

All detected optimal PS-networks are of "array-like" type: They consist of series configurations linked in parallel where the numbers of components in the series configurations differed at most by one.

Some future work needs to the done: The obvious next step should be to remove the restriction to 2-stage networks and consider multi-stage networks instead. Page and Perry have addressed this situation in their stimulating article [5], but their strategy of complete enumeration only works for small n. One could possibly apply well-known techniques of global combinatorial optimization, such as taboo search or simulated annealing, to this problem. Initial solutions, as they are required by such techniques, could be obtained by restriction to the 2-stage case and application of the above presented algorithm: It was remarked in [5] that not in all, but in several cases the optimal solutions even of the multi-stage problem were arrays.

Acknowledgment

We are indebted to V. Zaslavskii for drawing our attention to the problem investigated here.

References

[1] F.S. Hillier, G.J. Liebermann, *Introduction to Operations Research*, Holden-Day (1980).

[2] N.G. De Bruijn, "Denumerations of rooted trees and multisets", *Discrete Applied Math.*, 6, 1983, pp. 25-33.

[3] B.W. Jenny, D.J. Sherwin, "Open and short circuit reliability of systems of identical items", *IEEE Trans. Reliability*, Vol. R-35, 1986, pp. 532-538.

[4] A. Nijenhuis, H.S. Wilf, *Combinatorial Algorithms*, Academic Press (1975).

[5] L.B. Page, J.E. Perry, "Optimal 'series-parallel' networks of 3-state devices", *IEEE Trans. Reliability*, Vol. R-37, 1988, pp. 388-394.

[6] J. Riordan, *An Introduction to Combinatorial Analysis*, Wiley (1958).

[7] V. Volkovich, A. Voloshin, V. Zaslavskii, I. Ushalov, *Models and Techniques for Optimization of Reliability of Complex Systems*, Naukovo Dumko (1993) (in Russian).

[8] V.A. Zaslavskii, O.V. Franchuk, "Optimal reservation of a complex system by two types of failure state elements", in: Vestnik K.G.U., *Modeling and Optimization of Complex Systems*, No. 8, pp. 64-68 (1989) (in Russian).