European Journal of Operational Research
Volume 12, Issue 1, January 1983, Pages 36–81

The simple plant location problem: Survey and synthesis

Jakob KRARUP
DIKU, Institute of Datalogy, Univerisity of Copenhagen, DK-2200 Copenhagen N, Denmark

Abstract

With emphasis on the *simple plant location problem* (SPLP), we consider an important family of *discrete, deterministic, single-criterion, NP-hard*, and widely applicable optimization problems. The introductory discussion on problem formulation aspects is followed by the establishment of relationships between SPLP and set packing, set covering and set partitioning problems which all are among those structures in integer programming having the most wide-spread applications. An extensive discourse on solution properties and computational techniques, spanning from early heuristics to the presumably most novel exact methods is then provided. Other subjects of concern include a subfamily of SPLP's solvable in polynomial time, analyses of approximate algorithms, transformability of *p*-CENTER and *p*-MEDIAN to SPLP, and structural properties of the SPLP polytope. Along the way we attempt to synthesize these findings and relate them to other areas of integer programming.

Introduction

The past two decades have witnessed an explosive growth in the literature on location problems. This is not at all surprising since locational decisions is one of the more profitable areas of applied O.R. and ample theoretical challenges are offered. However, among the myriads of formulations considered, only four of these: *p*-CENTER, *p*-MEDIAN, SIMPLE PLANT LOCATION, and QUADRATIC ASSIGNMENT – at times referred to as *prototype location problems* – have played a particularly dominant role. If seminal works such as Fermat's 1-MEDIAN from the early 1600's and Sylvester's 1-CENTER brain teaser from 1857 are disregarded, all four problems entered the stage in their present form in the period 1957–64.

In contrast to *p*-CENTER and *p*-MEDIAN treated at length in textbooks including Francis and White (1974), Christofides (1975), Jacobsen and Pruzan (1978), Handler and Mirchandani (1979) and in the survey by Krarup and Pruzan (1979), we have long sought in vain for a comprehensive exposition with particular focus on the simple plant location problem (SPLP). This is surprising since, judging from a rough estimate of the number of papers devoted to each prototype problem and from their applicability to real-life decision-making, SPLP seems to have attracted most attention. In chronological order, early reviews and shorter summaries of the state of the art can be found in Balinski and Spielberg (1969), ReVelle et al. (1970), Eilon et al. (1971), Hansen (1972), Elshafei and Haley (1974), Francis and White (1974), Kaufman (1975), Salkin (1975), Jacobsen (1977), Guignard and Spielberg (1977), Jacobsen and Pruzan (1978) and Cornuéjols (1978).

The authors are indebted to Egon Balas, Jens Clausen, Monique Guignard-Spielberg, the late Jonathan Halpern, Antoon Kolen, Kurt Spielberg, Philip Wolfe, and Stanislaw Walukiewicz for valuable suggestions on the preliminary version of this paper. We also wish to express our gratitude to the referees, James G. Morris and Richard M. Soland, who were so kind as to divulge their identities. The final version has benefited from their meticulous screening and constructive criticism.

North-Holland Publishing Company
European Journal of Operational Research

0377-2217/83/0000-0000/$03.00

The present exposition is an attempt to fill the gap. To give adequate credit to all previous contributors would be a Herculian task, far beyond our level of ambition; instead, we shall consider some of the main findings, spanning the SPLP-literature from what appears to be the first formulation of the problem in the early 60's to the most recent developments, represented by a series of papers currently under publication. Along the way we shall attempt to synthesize these findings, relate them to other areas of general integer programming and suggests likely courses of future developments.

While most well-defined problems bear unambiguous names, SPLP has been dealt with in the literature under a wide variety of different titles usually composed of an adjective (uncapacited, simple, optimal) and a substantive (plant, warehouse, facility, site) followed by the word location. Including omission of the adjective, we have among the possible combinations seen ten or so of these employed so far. And new contributions to the name confusion arise if 'economies-of-scale' or 'fixed-cost discrete space location-allocation' and the like are incorporated, not to speak of the flock of variations, based upon word-to-word translation into English of non-English designations. Even some familiarity with certain non-English languages does not always suffice for identifying a SPLP-paper written in one of these via its title.

SPLP derives its name from the analogy to decision problems concerning the location of *plants or facilities* (e.g. factories, warehouses, schools) so as to minimize the total cost of serving *clients* (e.g. depots, retail outlets, students). We postulate that the major reason that SPLP has been the subject of so much attention is that despite (or perhaps due to) its transparent structure, it has contributed to the formulation and solution of a multitude of complex planning problems. Unfortunately, to the best of our knowledge there does not exist a readily available list of such applications which could lend credence to our postulate; we can though draw upon our own experience as consultants where we have utilized SPLP formulations as the basis for providing decision inputs to real-world problems regarding the number, size, design, location, and service patterns for such widely varied 'plants' as high-schools, hospitals, silos, slaughterhouses, electronic components, warehouses, as well as traditional production plants.

In contrast to the other prototype location problems, SPLP permits in a sense the broadest framework. Neither the number of plants to be located nor the transportation or communication pattern are predetermined. Furthermore, the basic formulation of SPLP lends itslef readily to sensitivity analyses. In addition, SPLP invites modifications which may permit more 'realistic' modelling. While SPLP is basically a *discrete, static, deterministic, one-product, fixed-plus-linear costs minimization problem formulation,* it can be modified to accommodate *dynamic, stochastic, multi-product, nonlinear cost minimization formulations.* It can also include *prices* as well as costs in its criteria and can also be used within the context of *multicriteria optimization.* Last, but not least, there exists a veritable arsenal of well documented exact algorithms, heuristics, relaxations, and simulation procedures, more or less tailored to provide solutions to SPLP's.

It appears thus that the attention paid to SPLP is due to both its relevance for sundry decision problems, its simple, easily grasped structure, and the availability of effective solution methods. There is however the reservation that while p-CENTER and p-MEDIAN usually are dealt with in both continuous (planar) and discrete (network) formulations, there is a dearth of literature as to formulations of SPLP in the plane. We shall therefore be mainly concernced with the SPLP-family within the context of *networks* but briefly comment the very rare exceptions in Section 13.

Our presentation of SPLP and its connections to close and more distant relatives is organized as follows: Alternative symbolic formulations of SPLP are introduced and their properties are briefly discussed in Section 1; furthermore, the blurred historical origin and some notational discrepancies are accounted for. The relationships between SPLP and set covering, set packing, and set partitioning problems are established in Section 2, emphasizing once more the applicability of SPLP, while Section 3 provides a rudiment of computational complexity to demonstrate that SPLP (as could be expected) belongs to the class of notorious optimization problems which is known as NP-hard. The five following sections deal with solution methods and computational aspects. Like other difficult combinatorial problems, it is not surprising that SPLP was only amenable to heuristics during the first years of its lifetime; a collection covering the period up to 1966 is reviewed in Section 5. Numerous exact methods have appeared since 1966, culminating with the dual-based algorithms, rooted in Lagrangean relaxation, and claimed to be 'close to the ultimate in efficiency'. The extensive selection presented in Sections 6–8 demonstrates the

diversification of integer programming techniques adapted for SPLP over the years. We proceed in Section 9 with a class of specially structured SPLP's solvable in polynomial time, and capable of modelling certain 'classical' production planning problems. Analyses of heuristics and relaxations for a composite formulation having both p-MEDIAN and SPLP as special cases were initiated by Cornuéjols et al. (1977a) under the heading ACCOUNT LOCATION and have since then enjoyed much fruitful research. Some of the main results are summarized in Section 10 together with an observation due to Krarup and Pruzan (1981b) stressing the need for further refinements of error analysis. Based on Krarup and Pruzan (1981c), where convex combinations of p-CENTER, p-MEDIAN, and SPLP are integrated in a single model, it is shown in Section 11 that p-CENTER and p-MEDIAN are transformable to SPLP. Structural properties of the SPLP polytope are considered in Section 12 where certain families of facets are identified. Section 13, a mixture between an appetizer and an annotated bibliography, deals with selected variants and extensions of SPLP; pertinent keywords are capacitated, piecewise linear, dynamic, stochastic, continuous, multicommodity, multicriteria, investment and pricing.

1. Problem formulation

Given a finite set of possible locations for establishing new facilities or redimensioning already existing facilities, SPLP deals with the supply of a single commodity (or standard productmix) from a subset of these to a set of clients with a prescribed demand for the commodity. Facilities are assumed to have unlimited capacity such that in principle any facility can satsify all demands. For given costs associated with the facilities and with the direct transportation routes from facilities to clients, we seek a minimum cost production/transportation plan (in terms of the number of facilities established, their locations, and the amount shipped from each facility to each client) satisfying all demands. The constituents of SPLP are:

m : The number of potential facilities indexed by i, $i \in I = \{1,\ldots,m\}$,
n : The number of clients indexed by j, $j \in J = \{1,\ldots,n\}$,
f_i : The fixed cost of establishing facility i,
p_i : The per unit cost of operating facility i (including variable production and administrative costs etc.),
b_j : The number of units demanded by client j,
t_{ij}: The transportation cost of shipping one unit from facility i to client j.

It is customary to use the adjectives 'open' and 'closed' for designating the state of a facility. The cost of sending no units from a facility is zero (i.e. the facility is closed) while any positive shipment from the ith facility incurs a fixed cost f_i (the facility is open) plus costs $p_i + t_{ij}$ per unit produced at facility i and transported to the client j. We introduce the $m + mn$ variables

y_i : $y_i = 1$ if facility i is open and 0 otherwise,
s_{ij}: Number of units produced at facility i and shipped to client j.

The full-blooded SPLP is the mixed-integer program:

$$\min \sum_{i\in I}\sum_{j\in J}(p_i + t_{ij})s_{ij} + \sum_{i\in I} f_i y_i, \tag{1}$$

$$\sum_{i\in I} s_{ij} \geq b_j, \qquad j \in J, \tag{2}$$

$$k_i y_i - \sum_{j\in J} s_{ij} \geq 0, \quad i \in I, \tag{3}$$

$$s_{ij} \geq 0, \qquad i \in I, j \in J, \tag{4}$$

$$y_i \in \{0,1\}, \qquad i \in I. \tag{5}$$

The m restrictions (3) are devices to ensure that the total fixed cost for a facility is incurred whenever

positive shipments are made from it. The k's are positive constants, not less than the maximal outflow from the corresponding facilities. If all $p_i \geq 0$, $t_{ij} \geq 0$, no facility need ever ship more than the total amount demanded, and each k_i may be replaced by $\sum_{j \in J} b_j$. Similarly, since under the assumption $(p_i + t_{ij}) \geq 0$ it will never pay to ship a larger number of units to a client than demanded, the inequalities (2) can be replaced by equations.

Should negative $(p_i + t_{ij})$ occur for a given data instance, we can by means of appropriately selected constants perform a transformation so that the assumption above of nonnegativity holds: replace $(p_i + t_{ij})$ by $(p_i + t_{ij}) + \alpha_j$ all i, j where $\alpha = \{\alpha_j\}$ is an n-vector of constants such that $|\alpha_j| \geq \max_{i \in I} \{|p_i + t_{ij}| | (p_i + t_{ij}) < 0\}$. This transformation will not affect the optimal solution to the given instance since the only effect will be that the objective function is increased by the constant term $\sum_{j \in J} \alpha_j$.

Similarly, should any f_i be negative which, from a practical viewpoint, is rather absurd, it pays to open the ith facility irrespective of positive outflow or not. In such cases, f_i is added to the objective function (which hereby is reduced by a constant) and f_i is replaced in (1) by zero. Without loss of generality (and with the real-world applications in mind) we shall therefore assume all f_i to be nonnegative.

Note that the formulation remains unaltered if we modify the definitions of f_i and b_j by the words 'per time period'. In this case the fixed costs will include per time period investment costs and fixed operating costs corresponding to the minimal costs required to maintain an open facility.

Given the cost structure for SPLP, the decision problem is only considered to involve the flows (the s_{ij}'s) instead of both flows and facility *design*. The costs are assumed to adequately reflect the type, size, and location of the facilities and to only depend upon whether or not flow occurs from a facility. Were the costs also dependent upon the type and size of the facility to be located, the total costs could not be determined from the flow alone but would have to involve decision variables corresponding to the type and size of each facility as well. This subject is reconsidered in Section 13 where piecewise linear cost functions are discussed.

For any given binary y-vector an optimal set of s_{ij}'s is easily determined. Let $P = \{i | y_i = 1\}$ be a nonempty subset of open facilities and let $P(j) = \{i \in P | (p_i + t_{ij}) = \min_{k \in I} \{p_k + t_{kj}\}\}$. If for all j, $s_{ij} := b_j$ for some $i \in P(j)$ ($:=$ means 'is set equal to' and 'some' indicates that ties are resolved arbitrarily) and $s_{ij} := 0$ otherwise, we have evidently found an optimal production/transportation plan with respect to the given set P simply by assigning each client to the 'cheapest' open facility. In this sense, the y's can be claimed to constitute the essential variables of the problem, and for that reason some authors prefer to distinguish between the *strategic* variables (y_i) and the *tactical* variables (s_{ij}). Accordingly, the number of distinct solutions (in terms of the strategic variables) amounts to $2^m - 1$, the number of combinations of open and closed facilities, excluding the case where all facilities are closed. Note for a comparison that the total number of distinct assignments of clients to facilities (each client is served by, or assigned to exactly one facility) is m^n.

How trivial it may seem, the fact that we only have to consider solutions where every client is supplied by a single facility, plays a significant role for several of the computational methods to be presented. To stress its importance, we state it formally:

Proposition 1. *SPLP has the Single Assignment Property.*

Based on this observation, some simplification of the formulation almost suggests itself. Let $c_{ij} = b_j(p_i + t_{ij})$ be the *total cost* for supplying all of client j's demand from the ith facility, and let x_{ij} denote the *fraction of client j's demand* supplied from that facility. As discussed previously, all c_{ij} can be assumed nonnegative.

For a given subset $P \subseteq I$ of open facilities, the total costs incurred for an optimal solution with respect to P is $\sum_{j \in J} \min_{i \in P} c_{ij} + \sum_{i \in P} f_i$. SPLP is then the problem of identifying a subset P minimizing the total costs. The following version may thus be referred to as:

SPLP: Combinatorial formulation

$$\min_{P \subseteq I} \left\{ \sum_{j \in J} \min_{i \in P} c_{ij} + \sum_{i \in P} f_i \right\}. \tag{6}$$

Upon inclusion of the additional constraint $|P| = p$, (6) is transformed to a model which for $I = J$ and $f = 0$ is the p-median problem in a network with vertex set I:

p-MEDIAN: Combinatorial formulation

$$\min_{P \subseteq I, |P| = p} \left\{ \sum_{j \in I} \min_{i \in P} c_{ij} \right\}$$

where c_{ij} usually is interpreted as the (weighted) distance between vertices i and j.

Both SPLP and p-MEDIAN are special cases of a composite model known as Account Location which plays a key role for the algorithmic analyses exposed in Section 10.

For a given SPLP at least one facility has to be open in any feasible solution (y, x); feasibility is assured by the condition $\prod_{i \in I}(1 - y_i) = 0$. By introducing $m^2 n$ auxiliary, binary variables u_{ikj} defined for all i, $k \in I, j \in J$ by

$$u_{ikj} = \begin{cases} 0 & \text{if } (c_{ij} < c_{kj}) \text{ or } (c_{ij} = c_{kj} \text{ and } i \leqslant k), \\ 1 & \text{otherwise} \end{cases}$$

we obtain the following formulation of SPLP:

SPLP: Pseudo-boolean formulation

$$\min \quad \sum_{i \in I} \sum_{j \in J} c_{ij} y_i \prod_{k \in I} (1 - y_k u_{ikj}) + \sum_{i \in I} f_i y_i;$$

$$\prod_{i \in I} (1 - y_i) = 0, \quad y, u = 0 \text{ or } 1.$$

For L sufficiently large, Hammer (1968) replaces the single constraint by the term $L \prod_{i \in I}(1 - y_i)$ added to the objective function. SPLP is then reduced to minimizing an unrestricted, real-valued function of binary variables.

As to a mixed-integer formulation resembling (1)–(5), there are other ways of linking the fixed costs to positive shipments than by means of (3). To obtain the desired effect, we can simply require $y_i - x_{ij} \geqslant 0$, all i, j. Another possibility is to introduce a large positive number L assuming that prohibited or blocked transportation routes (i,j), if any, correspond to $c_{ij} = L$. Let Q_i be the subset of indices referring to clients which can be supplied from facility i and let $n_i \leqslant n$ be the number of clients in that subset, i.e. $Q_i = \{j \mid c_{ij} < L\}$, $n_i = |Q_i|$. Two alternative formulations of SPLP can now be stated as the (mixed) 0–1 programs:

SPLP with disaggregated (strong) or aggregated (weak) constraints

$$\min \quad \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} + \sum_{i \in I} f_i y_i, \tag{7}$$

$$\sum_{i \in I} x_{ij} = 1, \qquad j \in J, \tag{8}$$

$$\textit{either} \quad y_i - x_{ij} \geqslant 0, \qquad i \in I, j \in J \quad (\textit{disaggregated or strong}), \tag{9a}$$

$$\textit{or} \quad n_i y_i - \sum_{j \in J} x_{ij} \geqslant 0, \qquad i \in I \quad (\textit{aggregated or weak}), \tag{9b}$$

$$x_{ij} \geqslant 0, \qquad i \in I, j \in J, \tag{10}$$

$$y_i = 0 \text{ or } 1, \qquad i \in I. \tag{11}$$

In the sequel, SPLP-S and SPLP-W refer to the two formulations with Strong and Weak constraints respectively.

Whether (9a) or (9b) is employed does not affect the optimal solution. However, several computational procedures are based on an *LP-relaxation* in which the integrality constraints (11) are replaced by the less

restrictive requirements $y_i \geq 0$, $i \in I$. In such cases it is important to be aware of the effect of replacing the $m \times n$ 'disaggregated' inequalities (9a) by the 'aggregated' set of the m inequalities (9b). Advantages and drawbacks are discussed when we focus on the computational aspects of SPLP.

1.1. The origins of SPLP: Who was first?

The 'who was first – question' may seem of minor importance only, but the somewhat obscure history and the incorrect references made now and then by various authors justify comment.

Disregarding problem formulations such as that in Baumol and Wolfe (1958) which upon appropriate redefinition of terminology can be interpreted as including SPLP as a special case, the first explicit formulation of SPLP is frequently attributed to Balinski (1966) whose expository article on integer programming includes the mixed-integer formulation (7)–(11) with the strong constraints (9a). The paper was presented at the The IBM Scientific Symposium on Combinatorial Problems in March 1964 but remained unpublished until 1966. After some search, a copy was finally made available to us by Kolen (1978) who drew our attention to an even earlier source, a Mathematica report by Balinski and Wolfe (1963) referred to in Balinski (1966), Wolfe was then approached for a copy; his reply (Wolfe (1980)) includes the following remarks: "The paper by Balinski and myself, 'On Benders decomposition and a plant location problem', has disappeared. We indeed wrote it. We do not remember too well what was in it. Several years ago Balinski, noticing that it was not in his files, asked me for a copy; I didn't have it either. According to him, it also turned out not to be in Mathematica's official archives".

However, SPLP's are also dealt with in the pioneering papers by Kuehn and Hamburger (1963) and Manne (1964). Both are often cited and will deservedly be discussed in Section 5.

The *Journal of Farm Economics* is not exactly a traditional forum for locationists and discrete optimizers; accordingly, apart from researchers in the field of agricultural management, only a very few authors have noticed the seminal paper by Stollsteimer (1963). Based on the unpublished Ph.D. thesis, Stollsteimer (1961), he proposes four models of which the second actually is a SPLP. In a historical context, it should not detract from his contribution that the solution method devised is mere complete enumeration, thus allowing any cost structure whatsoever. On the contrary, judging from the annotated bibliography Lea (1973), in spite of its obvious weaknesses, Stollsteimer's work managed to generate a flock of Ph.D. theses and published papers on agricultural economics; see e.g. Ladd and Halvorson (1970), Chern and Polopolus (1970), and Warrack and Fletcher (1970), all discussed in some detail in the monograph, Elshafei and Haley (1974).

Thus, Hansen et al. (1981) seem pretty close to the truth by owing credit to Balinski, Manne and Stollsteimer for the first formulation of SPLP. Is the full truth then that SPLP was formulated independently by Stollsteimer, Balinski (and possibly Wolfe), Kuehn and Hamburger, and Manne? The lack of cross references between their works bears evidence to this hypothesis.

1.2. A comment on the terminology and notation employed.

SPLP can be viewed upon as a *covering problem*: by selecting an entry (i,j) corresponding to the transportation route from facility i to client j, we have in a sense *covered* the jth column by the ith row. However, it is customary in formulating mathematical programming problems that the right-hand side literally *is* on the right-hand side of the coefficient matrix; consequently, SPLP should rather be a problem of covering rows by means of columns. To conform to this commonly accepted rule, I and J should have been interchanged in the formulation. There seems however, to be a discrepancy between two different traditions: of all formulations of SPLP found in the literature, only a few are consistent with the notion of covering rows while the vast majority in principle apply the same notation as was adopted in (7)–(11). We felt tempted to join the first group (authors of future SPLP-papers are strongly urged to do so) but decided not to because of the confusion this would cause readers who will consult earlier works referred to here.

To avoid perpetual redefinitions of the most common concepts employed, we assume wherever appropriate that the three groups of identifiers $(I, i, m, f_i, p_i, Q_i, n_i, y_i)$, $(t_{ij}, s_{ij}, c_{ij}, x_{ij})$ and (J, j, n, b_j) are

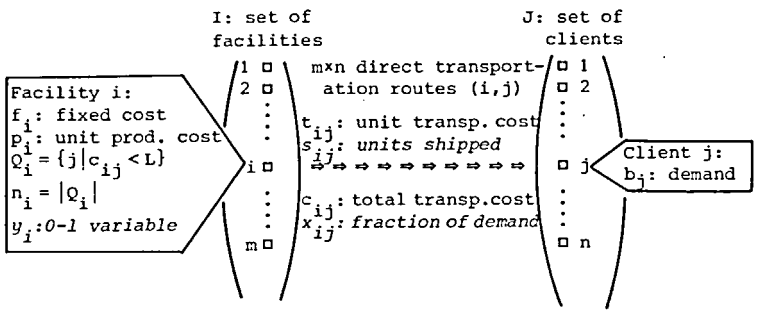


Fig. 1. Overview of 'standard' identifiers employed: Normal typing: Constants (data instance), *Italic typing: Variables.*

reserved for 'standard' purposes as visualized in Fig. 1.

Furthermore, to simplify the notation, we adopt Σ_i, $\max_i\{\ \}$, $y_i = 0$ or 1, all i, ... as shorter forms of $\Sigma_{i\in I}$ (or $\Sigma_{i=1}^{m}$), $\max_{i\in I}\{\ \}$, $y_i \in \{0,1\}$, $\forall i \in I$, ... respectively, provided that the meaning is clear from the context. Finally, for example, SPLP (10, 20) will refer to a data instance of SPLP with $m = 10$ and $n = 20$, similarly, SPLP $(2,n)$ comprises all data instances with $m = 2$.

2. SPLP and its relations to packing, covering and partitioning

Via seven transformations we shall establish relationships between SPLP and three of the most widely studied structures in (pure) integer programming: set packing, set covering and set partitioning. The purpose is twofold: to provide further evidence as to the postulated versatility of SPLP models in practice and to provide the background for a series of theoretical results with particular emphasis on computational complexity.

For finite sets I and J, let $A = \{a_{ij}\}$ be an $m \times n$ matrix of zeros and ones. To conform to the notational conventions set forth in the previous section, we shall in the sequel associate the variables with the rows of A. Thus, a subset $\bar{I} \subseteq I$ of rows defines a *cover* of J if $\Sigma_{i\in\bar{I}} a_{ij} \geq 1$, all j. $I^* \subseteq I$ is called a *partition* (exact cover) of J if equality holds for all j, i.e. if $\Sigma_{i\in I^*} a_{ij} = 1$. For all i, let $y_i = 1$ if row i is in the cover and 0 otherwise, and let f_i be the cost incurred for $y_i = 1$. The (weighted) set covering problem is to find a cover of minimum cost

$$\min \quad \sum_i f_i y_i,$$

$$\sum_i y_i a_{ij} \geq 1, \quad j \in J, \qquad y_i = 0 \text{ or } 1, \quad \text{all } i$$

or, in a more compact notation

SET COVER

$\min\{fy \mid yA \geq e, y_i = 0 \text{ or } 1, \text{all } i\}$

where e is an n-vector of ones. Accordingly, the (weighted) set partitioning problem reads

SET PARTITION

$\min\{fy \mid yA = e, y_i = 0 \text{ or } 1, \text{all } i\}$.

By reversing the inequality sign of the constraint set $yA \geq e$ in SET COVER and by changing the objective from minimization to maximization, we arrive at the (weighted) set packing problem

SET PACKING

$\max\{fy \mid yA \leq e, y_i = 0 \text{ or } 1, \text{all } i\}$.

The seven transformations T1–T7 relating SPLP to the three problems formulated above or converting SET PARTITION to either SET COVER or SET PACKING are visualized in Fig. 2. The transformations will later be summarized in Propositions 2–4 and illustrated by a numerical example.

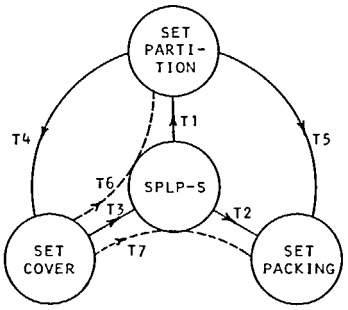


Fig. 2. Overview of transformations T1–T7.

In all cases involving SPLP, we will demonstrate the transformations based upon SPLP-S (SPLP with strong constraints). We shall first show how SPLP-S via simple transformations T1 and T2 can be brought to the form of SET PARTITION and SET PACKING respectively such that optimal solutions to these will also solve SPLP optimally.

Let $\bar{y}_i := 1 - y_i$, all i, and let r_{ij}, all i, j be nonnegative slack variables used to replace the disaggregated inequalities (9a) by equations. Now, SPLP-S becomes

$$\min \quad \sum_i \sum_j c_{ij} x_{ij} - \sum_i f_i \bar{y}_i + \sum_i f_i,$$

$$\sum_i x_{ij} = 1, \quad \text{all } j, \qquad x_{ij} + \bar{y}_i + r_{ij} = 1, \quad \text{all } i, j,$$

$$x_{ij}, r_{ij} \geq 0, \quad \text{all } i, j, \qquad \bar{y}_i = 0 \text{ or } 1, \quad \text{all } i.$$

Without affecting the optimal solution, we can explicitly require all x_{ij} to be 0 or 1 since SPLP as noted in Proposition 1 has the Single Assignment Property.Thus, all r_{ij} will be either 0 or 1 as well. The SPLP ⇒ SET PARTITION transformation follows directly from the definitions of $\bar{y}_i$ and r_{ij}:

T1: SPLP ⇒ SET PARTITION

$$\min \quad \sum_i \sum_j c_{ij} x_{ij} - \sum_i f_i \bar{y}_i \left(+ \sum_i f_i \right),$$

$$\sum_i x_{ij} = 1, \quad \text{all } j, \qquad x_{ij} + \bar{y}_i + r_{ij} = 1, \quad \text{all } i, j,$$

$$x_{ij}, r_{ij}, \bar{y}_i = 0 \text{ or } 1, \quad \text{all } i, j.$$

The term $\Sigma_i f_i$ in the objective function is shown enclosed in parentheses to indicate that it is constant.

As to transformation of SPLP-S to SET PACKING, we retain the definition of $\bar{y}$ as $\bar{y}_i := 1 - y_i$, all i. Furthermore, for all j, let w_j be a nonnegative 'artificial' slack variable used to convert the n equations (8) into inequalities, i.e. $w_j + \Sigma_i x_{ij} = 1$, $w_j \geq 0$ or $\Sigma_i x_{ij} \leq 1$, all j. To assure that all w_j become zeros in an optimal solution to SPLP, we associate with each of these slack variables a 'sufficiently large' weight L in the objective function. To obtain the desired effect, it suffices to choose $L > \max_j \{\min_i \{f_i + c_{ij}\}\}$. Since a solution x minimizing some function $g(x)$ will also maximize $-g(x)$, and since we, as before, can restrict all x_{ij}'s to be either 0 or 1, the objective function for SPLP becomes

$$\min \sum_i \sum_j c_{ij} x_{ij} - \sum_i f_i \bar{y}_i + L \sum_j w_j + \sum_i f_i$$

or, for $w_j = 1 - \Sigma_i x_{ij}$, all j

T2: SPLP $\Rightarrow$ SET PACKING

$$\max \quad \sum_i \sum_j (L - c_{ij}) x_{ij} + \sum_i f_i \bar{y}_i \left(-nL - \sum_i f_i \right),$$

$$\sum_i x_{ij} \leqslant 1, \quad \text{all } j, \qquad x_{ij} + \bar{y}_i \leqslant 1, \quad \text{all } i, j,$$

$$x_{ij}, \bar{y}_i = 0 \text{ or } 1, \quad \text{all } i, j.$$

As just shown, SPLP can be viewed as a highly structured special case of both SET PARTITION and SET PACKING in the sense that a substantial part of each coefficient matrix – as will be clearly seen from Example 1 – is an identity matrix.

To relate SPLP to SET COVER, consider the following specially structured SPLP where the ith facility (apart from the fixed cost incurred) can either serve or *cover* the jth client at no cost ($c_{ij} = 0$) or, if this is not the case, then it cannot serve that client at any finite cost ($c_{ij} = \infty$). Thus, to solve an instance of SPLP (m, n) for some f and with all $c_{ij} = 0$ or ∞ is merely to find a cover (i.e. a subset of facilities covering all clients) of minimum costs, that is, to solve the instance of SET COVER defined by m, n, A, f where $a_{ij} := 1$ if $c_{ij} = 0$ and $a_{ij} := 0$ otherwise.

This transformation (SPLP $\Rightarrow$ SET COVER) is based on the very rigorous assumption $c_{ij} = 0$ or ∞, all i, j, and will therefore not be included in the general results. However, the transformation SPLP $\Rightarrow$ SET COVER, valid for all data instances, will be established later via T1 and T4 as depicted in Fig. 2. On the other hand, the inverse transformation T3: SET COVER $\Rightarrow$ SPLP applies for any data instance m, n, A, f of SET COVER. If a feasible solution exists (clearly, a sufficient condition is that no column of A consists entirely of zeros) then an optimal solution to SET COVER can be found as an optimal solution to SPLP(m, n) with the given f-vector and $c = \{c_{ij}\}$ defined by $c_{ij} := 0$ if $a_{ij} = 1$ and infinite otherwise.

The following two transformations do not include SPLP explicitly but relate SET PARTITION to either SET COVER (T4) or SET PACKING (T5) respectively.

While any data instance of SET PACKING admits feasible solutions (e.g. a null vector or any unit vector), feasibility of a data instance for SET PARTITION is a rather intricate matter. Assume however that an instance of SET PARTITION defined by m, n, A, f is feasible. Then the following problems where $y_i = 0$ or 1 throughout, are equivalent in the sense that their optimal solutions coincide

$$\min \left\{ \sum_i f_i y_i \mid \sum_i y_i a_{ij} = 1, \text{ all } j \right\},$$

$$\min \left\{ \sum_i f_i y_i + L \sum_j w_j \mid -w_j + \sum_i y_i a_{ij} = 1, w_j \geqslant 0, \text{ all } j \right\}$$

for $L > \Sigma_i f_i$. We replace w_j by $-1 + \Sigma_i y_i a_{ij}$ in the objective function and obtain

$$\min \left\{ -Ln + \sum_i \left(f_i + L \sum_j a_{ij} \right) y_i \mid \sum_i y_i a_{ij} \geqslant 1, \text{ all } j \right\}$$

or

$$(-Ln) + \min\{(f + AeL) y \mid yA \geqslant e\}$$

which is SET COVER. Once again, as in T1 where a constant term $\Sigma_i f_i$ was included in the objective function, the objective function in T4 also includes a constant, here $-Ln$.

Another equivalent formulation of $\min\{\Sigma_i f_i y_i \mid \Sigma_i y_i a_{ij} = 1, \text{ all } j\}$ is

$$\min \left\{ \sum_i f_i y_i + L \sum_j w_j \mid w_j + \sum_i y_i a_{ij} = 1, w_j \geqslant 0, \text{ all } j \right\},$$

for $L > \Sigma_i f_i$ as before. Proceeding in the same manner as above, we obtain

$$\min \left\{ Ln + \sum_i \left(f_i - L \sum_j a_{ij} \right) y_i \mid \sum_i y_i a_{ij} \leqslant 1, \text{ all } j \right\}$$

or

$$(-Ln) = \max\{(-f + AeL)y \mid yA \leqslant e\}$$

which is SET PACKING.

Thus, for any data instance, m, n, A, f of SET PARTITION for which a feasible solution exists, an optimal solution can be found by solving either the instance of SET COVER defined by m, n, A, $AeL + f$ or the instance of SET PACKING defined by m, n, A, $AeL - f$ where $L > \Sigma_i f_i$ in both cases.

By showing the number of variables and constraints for both the original problem and its transformed version in each case, the five transformations developed so far can be summarized as

Proposition 2.

	Number of var.	constr.		*Number of* var.	constr.
T1: SPLP-S	$m + mn$	$n + mn$	$\Rightarrow$ *SET PARTITION*	$m + 2mn$	$n + mn$
T2: SPLP-S	$m + mn$	$n + mn$	$\Rightarrow$ *SET PACKING*	$m + mn$	$n + mn$
T3: SET COVER	m	n	$\Rightarrow$ *SPLP-S*	m	$n + mn$
T4: SET PARTITION	m	n	$\Rightarrow$ *SET COVER*	m	n
T5: SET PARTITION	m	n	$\Rightarrow$ *SET PACKING*	m	n

The relationships established via T1–T5 have, in some form or other appeared in the literature. The earliest publication year found for each is: T1: According to Guignard and Spielberg (1977), this transformation was discussed among Hoffman, Johnson and Padberg and suggested to the authors by Hoffman. T2: Padberg (1979); T3: Krarup (1967), T4: Lemke et al. (1971); T5: Balas and Padberg (1975b).

As to the transformation T4 in the inverse order, Balas and Padberg (1976) note that SET COVER cannot be brought to the form of SET PARTITION. Apparently, they tacitly assume that the number of variables and constraints should be preserved as was the case for T4 and T5; otherwise, if a (substantially) larger number of variables and constraints is permitted, it is well known (see Section 3) that the so-called decision problems underlying SET COVER and SET PARTITION can be transformed to each other (or for that matter to any other so-called NP-complete decision problem). What is noteworthy here is that the transformations SET COVER $\Rightarrow$ SET PARTITION or SET PACKING can be achieved via SPLP. More precisely, first T3 then T1 and T2 respectively:

Proposition 3.

	Number of var.	constr.		*Number of* var.	constr.
T6: SET COVER	m	n	$\Rightarrow$ *SET PARTITION*	$m + 2mn$	$n + mn$
T7: SET COVER	m	n	$\Rightarrow$ *SET PACKING*	$m + mn$	$n + mn$

Note that a data instance of SET COVER for which no feasible solution exists will correspond to an unbounded objective function for any feasible solution to the instances of SET PARTITION and SET PACKING as defined via SPLP by Proposition 3.

The following example illustrates all seven transformations:

Example 1. Let $(m, n) = (3,4)$ and

$$f = \begin{pmatrix} 3 \\ 4 \\ 2 \end{pmatrix}, \qquad A = \begin{Bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \end{Bmatrix}.$$

For this particular instance, let y^c and y^p denote optimal solutions to SET COVER and SET PARTITION respectively. It is seen that $y^c = (1, 0, 1)$, $fy^c = 5$, $y^p = (1, 1, 0)$ and $fy^p = 7$.

To save space we shall only consider the transformations T1 and T2 as parts of T6 and T7 respectively. The transformations are therefore presented in the order T3, T4, T5, T6 or T3 + T1, T7 or T3 + T2.

T3: SET COVER ⇒ SPLP: Let (x^0, y^0) be the optimal solution to the instance of SPLP defined by m, n, f, c, where $c_{ij} = 0$ if $a_{ij} = 1$; otherwise, $c_{ij} := \infty$:

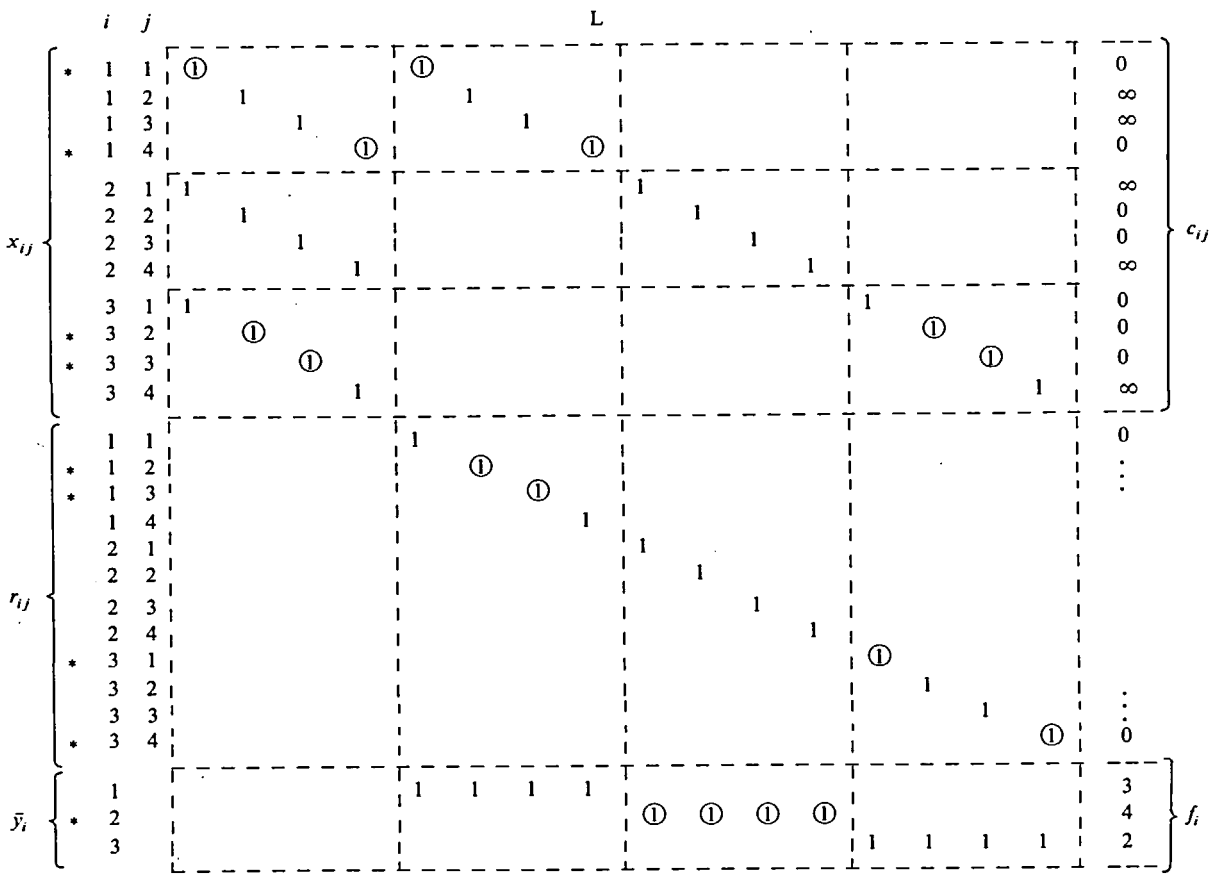
$$C = \begin{Bmatrix} 0 & \infty & \infty & 0 \\ \infty & 0 & 0 & \infty \\ 0 & 0 & 0 & \infty \end{Bmatrix}.$$

Thus, $y^0 = (1, 0, 1) = y^c$; x^0_{11} (or x^0_{31}) $= 1$; $x^0_{14} = x^0_{32} = x^0_{33} = 1$ while all other $x^0_{ij} = 0$. $\Sigma_i \Sigma_j c_{ij} x^0_{ij} + \Sigma_i f_i y^0_i = 5 = f y^c$.

T4: SET PARTITION ⇒ SET COVER: $L > \Sigma_i f_i = 9$; $L := 10$; $AeL = (20, 20, 30)$; $f + AeL = (23, 24, 32)$. Optimal solution y' to SET COVER with data m, n, A, $f + AeL$ is $y' = (1, 1, 0) = y^P$. We observe that $-Ln + (f + AeL)y' = -40 + 47 = 7 = f y^P$.

T5: SET PARTITION ⇒ SET PACKING: For $L = 10$, the optimal solution y'' to SET PACKING with data m, n, A, $-f + AeL$ or m, n, A, $(17, 16, 28)$ is $y'' = (1, 1, 0) = y^P$. We note that $-Ln + (-f + AeL)y'' = -40 + (17, 16, 28)y'' = -7 = -f y^P$.

T6 (T3 + T1): SET COVER ⇒ SPLP ⇒ SET PARTITION: The instance of SPLP defined by T3 is transformed to the instance of SET PARTITION shown below:

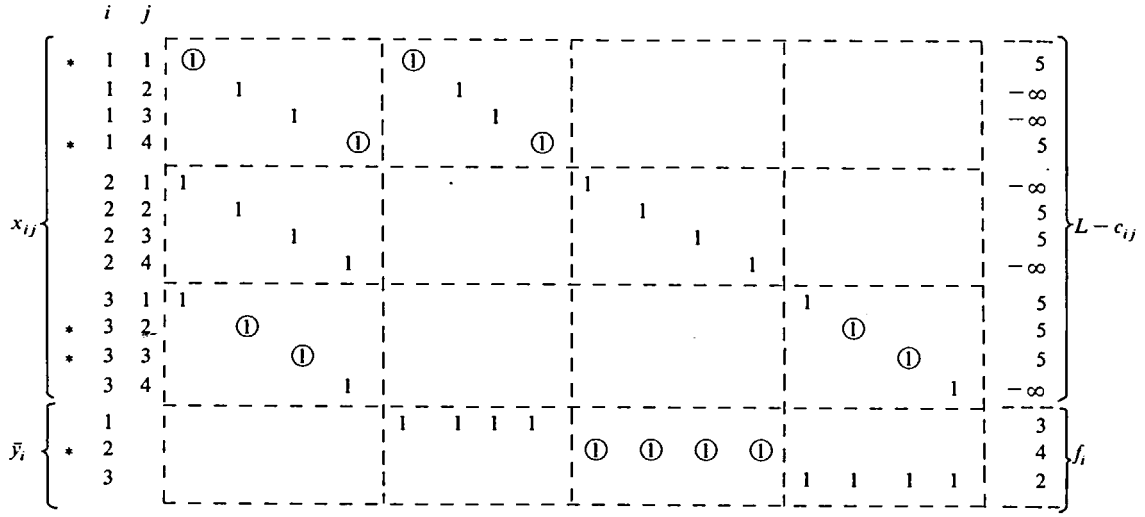


T6 (T3+T1): SET COVER ⇒ SPLP ⇒ SET PARTITION

The circled entries correspond to the starred variables equal to 1 in the optimal solution and clearly demonstrate that the solution found does in fact constitute a partition. The cost vector associated with the

$m + 2mn = 27$ variables is shown in the extreme right-hand column. For the solution shown, note that $\Sigma_i \Sigma_j c_{ij} x_{ij} - \Sigma_i f_i \bar{y}_i + \Sigma_i f_i = -4 + 9 = 5 = fy^c$.

T7 (T3 + T2): SET COVER ⇒ SPLP ⇒ SET PACKING: The same instance of SPLP is finally transformed to an instance of SET PACKING with $m + mn = 15$ variables. For $L > \max_j\{\min_i\{f_i + c_{ij}\}\} = 3$ or, for example, for $L := 5$, we obtain the instance of SET PACKING given below:



T7 (T3+T2): SET COVER⇒SPLP⇒SET PACKING

For the solution shown, $\Sigma_i \Sigma_j (L - c_{ij}) x_{ij} + \Sigma_i f_i \bar{y}_i - nL - \Sigma_i f_i = 5 \times 4 + 4 - 5 \times 4 - 9 = -5 = -fy^c$. □

Extensive use of transformations T1–T3 have been made in various contexts. Examples include studies of the SPLP-polytope discussed in Guignard and Spielberg (1977) and Guignard-Spielberg (1980), the polynomial heuristics devised in Hochbaum (1979), and the investigations of facets of the SPLP-polytope reported on in Cornuéjols and Thizy (1980) and discussed in Section 12.

In addition to the transformations shown, it will be established in Section 11 that p-MEDIAN and p-CENTER are transformable to SPLP such that the number of strategic y-variables is maintained.

3. The computational complexity of SPLP and related problems

According to Guignard and Spielberg (1977): "The SPLP is one of the simplest mixed integer problems which exhibits all the typical combinatorial difficulties of mixed (0–1) programming and at the same time has a structure that invites the application of various specialized techniques". This statement indicates that SPLP is a hard nut to crack, or, to use a more precise characterization, that it is highly unlikely that an exact polynomial time bounded algorithm can ever be devised for its solution.

In the following, we will provide a series of characterizations of the four related integer programming problems in terms of *computational complexity*, to demonstrate that they all indeed belong to the class of combinatorial optimization problems termed *NP-hard*.

Since excellent textbooks and expository articles on the theory of NP-completeness are available, we shall resort to intuitively appealing on definitions of the classes $\mathcal{P}$ and $\mathcal{NP}$, sufficient for conveying our main message: SPLP itself is termed NP-hard and the so-called *decision problem related to SPLP* is NP-complete. Pertinent references include Garey and Johnson (1979) and the survey-type paper by Lenstra and Rinnooy Kan (1979).

First a word about the distinction between polynomial and exponential time bounded algorithms. For our purposes, we can consider *algorithms* as step-by-step procedures for solving problems in a finite

computing time for all data instances. For a given problem type (e.g. SPLP) and a set of data instances of a given size corresponding to a given input length (e.g. SPLP (m, n)), the *complexity function* for an algorithm expresses the largest amount of time required for solving the problem for an arbitrary data instance of that size. An algorithm is called *polynomial* (*time bounded*) or *good* or *fast* if for all data instances its time complexity function is bounded by some polynomial function of the input length; otherwise the algorithm is called *exponential*. Since this definition involves 'all data instances', we can alternatively say that an algorithm is either polynomial or exponential in the *worst* (most time consuming) *case*.

In contrast to optimization problems dealt with so far, a *decision problem* π has only two possible solutions, either the answer 'yes' or the answer 'no'. As an example, a decision problem π (SPLP) related to SPLP is: For a given instance m, n, C, f and a given threshold value k, does SPLP have a solution of value at most k?

The formal theory of NP-completeness is based on the concept of deterministic and nondeterministic Turing machines, designed to provide yes/no answers to decision problems posed in terms of *language recognition*. For a given alphabet and language, an input consisting of a finite *string* of symbols from the alphabet is accepted by the machine (i.e. solves the decision problem) if and only if it belongs to the language.

We will consider a string as a *data instance* and a *language* as a *problem type* or as the *set of all its feasible instances*. A decision problem as to the feasibility of a data instance for a given problem type is said to belong to the class $\mathcal{P}$ if feasibility *or* infeasibility of any data instance can be determined by some algorithm in polynomial time on a digital computer, idealized to accept input of arbitrary length. Thus, $\pi(\text{SPLP}) \in \mathcal{P}$ if we for given m, n, C, f, k in polynomial time can either confirm or refute its membership of the set of all feasible instances. Actually, it is not known whether π(SPLP) in general belongs to $\mathcal{P}$ or not.

It does however belong to the (wider?) class $\mathcal{NP}$ which can be characterized without reference to the rather exotic concept of a nondeterministic Turing machine as follows: For a given instance γ of a decision problem (e.g. π(SPLP) defined by m, n, C, f, k) its feasibility implies the existence of an appropriate structure δ associated with γ (e.g. a binary vector of length $m + mn$ whose elements correspond to the assignment of values 0 or 1 to all variables y, x in (8)–(11)). If the length of δ (the encoding of the desired structure) is bounded by some polynomial in the length of γ and if we for given (γ, δ) can *affirm the feasibility* of δ (e.g. (8)–(11) are satisfied and the value of the objective function (7) does not exceed k) in polynomial time on a digital computer, this decision problem is then said to belong to the class $\mathcal{NP}$. This may also be informally expressed within a framework of nondeterministic computation by stating that $\mathcal{NP}$ is the class of all decision problems that can be solved by polynomial time, nondeterministic algorithms.

Evidently, $\pi(\text{SPLP}) \in \mathcal{NP}$ as asserted since both conditions for its membership are satisfied; the Single Assignment Property ensures that δ is of length $m + mn$ and to verify its feasibility requires computations of the order $m + mn$.

If for any data instance of a decision problem π' we can construct in polynomial time a data instance of a decision problem π such that the instance of π' is feasible if and only if the instance of π is feasible, then π' is said to be *polynomially transformable* to π (notation: $\pi' \propto \pi$). Thus, $\pi' \propto \pi$ implies that π' can be viewed as a special case of π and, consequently, π is at least as difficult to solve as π'. If $\pi' \propto \pi$ for all $\pi' \in \mathcal{NP}$ then any problem belonging to the class $\mathcal{NP}$ can be viewed as a special case of π which is then called *NP-hard*. Finally π is called *NP-complete* if π is NP-hard and $\pi \in \mathcal{NP}$. This definition of NP-completeness does not provide an efficient basis for proving that a decision problem π is NP-complete. However, it can be shown that if some $\pi' \propto \pi$, π belongs to $\mathcal{NP}$ and π' is NP-complete, then π is also NP-complete.

The Main Theorem in Karp (1972) asserts the NP-completeness of 21 decision problems including π(SET COVER). If we for a given instance, m, n, A, f, k of π(SET COVER) define an instance of SPLP as shown earlier (Proposition 2, transformation T3) then the answer to π(SET COVER) will be positive if and only if the answer to the question: "Does this instance of SPLP have a solution of value at most k?" is positive. Since π(SPLP) is in $\mathcal{NP}$ and since the transformation $\pi(\text{SET COVER}) \Rightarrow \pi(\text{SPLP})$ is polynomial (in this case linear) in mn then π(SPLP) is NP-complete.

Our primary concern in this paper is certain *optimization* problems and their computational tractability

whereas the temporary interest in decision problems is due merely to the fact that key concepts like NP-completeness and NP-hardness are only defined with respect to these. What remains in order to characterize the computational complexity of SPLP itself is therefore to elaborate upon the connection between a given optimization problem and *some* related decision problem. For a given minimization problem OPT, the most direct way of deriving a related decision problem π_0(OPT) is to introduce a *threshold* k and ask the question: "Does OPT have a feasible solution of value at most k?". If π_0(OPT) – which for given OPT and given k is uniquely defined – possesses a certain property (e.g. NP-completeness) and if this property is retained upon certain simplifications of π_0(OPT) then alternative decision problems π_1(OPT),... possessing that property can be formulated. Consider e.g. OPT = SET PARTITION. For a given data instance m, n, A, f, k, we have π_0(SET PARTITION): "Does there exist a y which is feasible for SET PARTITION and satisfies $fy \leqslant k$?". A simplification preserving the NP-completeness of π_0 is to consider the *unweighted* case where all $f_i = 1$: π_1(SET PARTITION): "Does there exist a y which is feasible for SET PARTITION and which satisfies $\Sigma_i y_i \leqslant k$?". This decision problem can be shown to remain NP-complete even if we drop the threshold: π_2(SET PARTITION): "Does there exist a feasible y?".

In order to characterize the computational complexity of optimization problems, a π(OPT) related to a given OPT is usually chosen as the most simplified version, simplified in the sense that its complexity remains unaltered, as exemplified by π_2(SET PARTITION) above. Note that a simplification similar to that from π_1 to π_2 such that the decision problem remains NP-complete does not apply for π(SET COVER), π(SET PACKING) and π(SPLP). With k disregarded the answer 'yes' or 'no' as to the feasibility of a data instance for these decision problems can be provided in polynomial time.

For any instance of a decision problem $\pi \in \mathcal{P}$ answers can be provided to both the question "is this instance feasible" and the complementary question "is this instance infeasible". However, for $\pi \in \mathcal{NP}$, a similar symmetry between a problem and its complement is not known to exist since in general for a given instance only the *feasibility* of an associated structure can be *affirmed* in polynomial time. Actually no complement of any NP-complete problem (e.g. for a given instance m, n, C, f, k of π(SPLP): "Is it true that *no* feasible solution is of value at most k?") is known to belong to $\mathcal{NP}$.

An instance of a given OPT cannot in general be optimally solved in polynomial time by solving a sequence of some related π(OPT) for varying thresholds. The assertion is true in general even for an optimization problem OPT whose corresponding decision problem π(OPT) $\in \mathcal{P}$. Though the infeasibility as well, as the feasibility of an instance of π(OPT) can be determined in polynomial time, the number of such instances to be solved will not in general be bounded by some polynomial in the length of the input. Furthermore, since $\mathcal{P} \subseteq \mathcal{NP}$, the assertion holds true for π(OPT) $\in \mathcal{NP} \setminus \mathcal{P}$ as well be it empty or not. Consequently, membership of $\mathcal{NP}$ for some π(OPT) does not necessarily imply that OPT itself, viewed as a sequence of decision problems, belong to $\mathcal{NP}$.

We are thus unable formally to prove, say, that SPLP is NP-complete or NP-hard, whereas the following argument provides at least a reasonable characterization of its computational complexity. Evidently while the converse is not true, optimal solution of some OPT will solve a related decision problem π(OPT) as well. Furthermore, if OPT is solvable in polynomial time then so is π(OPT), and if π(OPT) in addition is known to be NP-hard, then any decision problem in $\mathcal{NP}$ is solvable in polynomial time and $\mathcal{P} = \mathcal{NP}$. Since NP-hardness is defined only with respect to decision problems, then *the corresponding optimization problems* can be called *NP-hard* in the sense that the existence of polynomial algorithms for their solution would imply $\mathcal{P} = \mathcal{NP}$.

Although π(SET PACKING) has not been explicitly stated, it is easily verified via the polynomial transformations shown that the discussion on complexity can be summarized as

Proposition 4.

π(SET COVER) $\propto$ π(SPLP).

π(SPLP) $\propto$ π(SET PARTITION).

π(SPLP) $\propto$ π(SET PACKING).

π(SET COVER) is NP-complete $\Rightarrow$ π(SPLP), π(SET PARTITION) and π (SET PACKING) are all NP-complete.

The four corresponding optimization problems are called NP-hard.

Although the theory of computational complexity has provided useful complexity measures of decision problems via the definition of classes $\mathcal{P}$ and $\mathcal{NP}$ followed up by concepts like NP-completeness and NP-hardness, the fact that some NP-hard optimization problems in practice (i.e. not in the 'worst-case' situations underlying complexity analysis) are computationally more demanding than others is still a constant source of dissatisfaction. There is thus a strong need for further refinements of complexity measures.

4. Exact and approximate algorithms: A brief discourse

For a given data dependent set S of feasible solutions to an optimization (minimization) problem and a given, real-valued, data dependent function $g(x)$, we shall initially classify algorithms for finding either $z^0 = g(x^0) = \min\{g(x) | x \in S\}$ or some 'reasonable' solution x'.

For a given problem type and *for all data instances*, assume that the algorithm terminates after a finite computing time with a solution x' and the corresponding value $z' = g(x')$ of the objective function. If $x' \in S$ and $g(x') = z^0$, the algorithm is called *exact*; otherwise, it is *approximate*. If $x' \in S' \supseteq S$ and $z' \leqslant z^0$, some of the constraints are *relaxed* and x' need not be feasible with respect to the original set S. In this case the approximation algorithm is said to solve a *relaxation* of the original problem. Note that although the word 'relaxation' refers to the reduced constraint set, some authors find it practical to use 'relaxation' as synonymous with an approximation algorithm which provides solutions possessing the above properties. Finally, if $x' \in S$ and $z' \geqslant z^0$, the approximation algorithm is a *heuristic*. A heuristic is thus an algorithm producing feasible solutions for all data instances without guaranteeing optimality.

It is practical and commonly accepted to characterize algorithms by a measure related to their time complexity function. For a specific problem type OPT and a specific algorithm A, the time complexity function $f(n)$ will, as mentioned in Section 3, express the largest amount of time required for solving any instance of length n. To characterize the order of $f(n)$, we say that $f(n)$ is $O(g(n))$ whenever some constant α exists such that $|f(n)| \leqslant \alpha |g(n)|$ for all nonnegative n. We then refer to the algorithm A as an $O(g(n))$ algorithm for OPT. To exemplify this notation, for some OPT, A and for any positive n, let e.g. $f(n) = 7n^5 + 3n^2 + \log n$. For $g(n) = n^5$, $|f(n)| \leqslant 10|g(n)|$, hence, A is an $O(n^5)$ algorithm for OPT.

For OPT = SPLP, a 'data instance of length n' has no obvious meaning since a data instance m, n, C, f consists (at most) of $1 + 1 + mn + m$ numbers. We can however introduce $t = \max\{m, n\}$ whereby the length of any data instance will be at most a polynomial of second order in t. Since the product of two polynomials again is a polynomial, any algorithm which is $O(p(t))$ for some polynomial p of $t = \max\{m, n\}$ is also polynomial time bounded for SPLP(m, n). This corresponds to characterizations of (approximate) algorithms for TRAVELING SALESMAN like 'an $O(n^3)$ algorithm' where n traditionally refers to the number of cities and not to the length of a data instance which is of the order n^2 (n itself plus an $n \times n$ matrix of intercity distances).

As to algorithms devised since the mid 60's for SPLP, the inventiveness has been great. A wide selection of different mathematical programming techniques have with varying degrees of success been tailored for SPLP, and numerous computer codes have been written, implemented, tested, compared, and reported on in the literature. In addition to more or less sophisticated approximation algorithms, we have found exact methods based on cutting planes, dynamic programming, pseudo-boolean programming plus a long list of other enumerative approaches with a combination of LP-relaxation and branch-and-bound as the most successful recorded to data.

Like earlier approaches for other hard combinatorial optimization problems, it is only natural to expect the first methodologies for solving SPLP to be built on heuristics. At a time when more advanced tools were not yet developed to a satisfactory level, most O.R. practitioners were compelled to resort to heuristics in the hope that a sound philosophy behind such an approach would lead to a reasonably good result. The decision-maker's acceptance of the solution presented to her was often (and still is) regarded as an adequate quality measure, or possibly the only one within reach.

The selected samples to be considered in the sequel are all outdated in the sense that they are

outperformed (with respect to quality of solutions, computational requirements and the size of data instances amenable to solution) by more recent techniques. Nevertheless, these 'ancient' heuristics should not be totally neglected. Besides representing landmarks on the road from past to present, there seems to be a permanent need for heuristics either to achieve computational economy or to approach problems characterized by incomplete data or by a dimension beyond the capability of existing exact algorithms.

It has not always been fashionable for academics to devote time to analyses of the quick and dirty methods characterizing much practical O.R. work, apparently because of their often postulated lack of mathematical elegance. However, with the advent of recent results of analyses of approximate algorithms, this attitude has completely changed. Judging from the latest contributions to the theory of combinatorial optimization, hardly any other subject appears to attract more attention than studies of worst-case and average-case performance of heuristics and relaxations.

While the following section will deal only with heuristics without providing such analyses, an account of preliminary results for SPLP-related algorithms is given in Section 10.

5. Early heuristics (1963–66)

One of the earliest methods proposed for SPLP is the now well-known heuristic due to Kuehn and Hamburger (1963). It has been widely studied and its catalytic effect can hardly be overrated. Although a far more comprehensive model (involving plants, intermediate warehouses, clients, several commodities, plant- and warehouse capacities, and allowing a very complex cost structure including a cost term for delays in delivery) is formulated, the 12 sample problems studied are all essentially reduced to SPLP (24,50). The heuristic consists of two parts. The main program is an 'add routine' whereby facilities are located one by one (corresponding to the greatest cost reduction) until no facility can be added without increasing total cost. The underlying hypothesis is that the optimal solution for $(p+1)$ facilities can be determined from the optimal solution for p facilities by adding an additional facility to the existing solution. In a modern terminology, such a scheme would be called *greedy* because of its appetite for maximum improvement at each step. Upon termination of the main program, a so-called bump and shift routine is entered. It first eliminates (bumps) any facility which is now uneconomical because of the proximity of another facility located subsequently. It also considers relocating (shifting) each facility from its actual location to other potential locations in its neighborhood. However, the improvements by the bump and shift routine never exceed 0.5 per cent for the 12 problems examined, thus lending support to the principles on which the main program is based. The computing time required for the 12 sample problems totals 72 minutes (IBM-650) and is reduced to one hour with a different version of the bump and shift routine. We will occasionally cite computing times due to their historical interest even though it is impossible to meaningfully compare times for different data instances run on different computers.

Manne (1964) provides a Steepest Ascent One Point Move Algorithm (SAOPMA) for SPLP, a heuristic proposed by Reiter and Sherman (1962) for a more general class of combinatorial optimization problems. SAOPMA is a greedy *improvement heuristic*, initiated with a feasible solution (in terms of the y_i's) which can be any of the 2^m lattice points of the unit hypercube excluding $(0, 0, \ldots, 0)$. It proceeds by moving to one of the m adjacent lattice points by replacing one of the y_i's with its complement, selected so as to give the greatest decrease in total cost thereby permitting both the addition of a new facility or the deletion of an existing facility. SAOPMA terminates with a suboptimal solution when movement to any adjacent lattice point will not result in a lower value of the objective function.

As pointed out by Feldman et al. (1966), the SPLP-formulation is inadequate for those problems where the economies-of-scale affect facility costs over the entire range of facility sizes. In SPLP the concave objective function is a result of the fixed costs only; consequently, a more general model is proposed:

$$\min\left\{\sum_i \sum_j c_{ij}s_{ij} + \sum_i f_i(s_i) \mid s_i = \sum_j s_{ij},\ \sum_i s_{ij} = b_j,\ \text{all } j,\ s_{ij} \geq 0,\ \text{all } i,j\right\}$$

where $f_i(s_i)$ is continuous and concave over the range of interest.

Should facilities be added or dropped? The KH-heuristic (Kuehn and Hamburger) is based on sequential addition of facilities, assuming that the best p facilities in general will be a subset of the best $p+1$. They note however that facilities might be eliminated (dropped) rather than added, i.e. facilities are in operation in every potential location in the first feasible solution and then eliminated one by one, guided by cost savings. Based on the Single Assignment Property which is easily verified to apply for problems with any concave facility costs, Feldman et al. modify the KH-heuristic by incorporating a 'drop-routine'. Their computational results on the KH sample problems show that the running time for each instance is reduced to under one minute (IBM-7094) and that no solution obtained has a higher cost than that observed by Kuehn and Hamburger. Note that SAOPMA considered above consists of both add and drop procedures, where an add or a drop can be performed at any step; in practice it usually performs like either an add or a drop procedure, dependent only on the choice of initial solution. In Jacobsen (1977) the three greedy one-point-move construction heuristics so far discussed are related to the dual-based and Lagrangean relaxation approaches considered in Section 8.

Another effort on the heuristic frontier is that of Bergendahl (1966). Since local minima can occur in nonconvex minimization, LP-techniques and the like (e.g. separable programming) can be expected to result in far-from-optimal local minima. Small-scale SPLP's due to Manne (1964) are resolved by separable programming, confirming this expectation. A modification called *Marginal Cost Parametrization* of the standard separable programming technique is therefore proposed, leading to better solutions to the sample problems than those obtained by Manne. However, the computing time is excessive even for problems of a very moderate size (complete enumeration is faster) so, in spite of the acceptable solutions found, separable programming-like techniques for SPLP and related problems cannot be recommended.

Apart from the separable programming heuristic, the three others will be revisited in Section 10, where the worst-case behaviour of several approximate algorithms is studied.

6. LP-relaxations of SPLP

LP-relaxations of SPLP have considerable interest as they provide the basis both for various approximation algorithms and for the determination of bounds for the most successful integer linear programming algorithms for SPLP (and for integer programming in general) based upon branch and bound type procedures. The following exposition characterizes such relaxations with respect to the quality of solutions and computational requirements.

As was mentioned in Section 1 in connection with the formulation of SPLP (7)–(11), alternative ways of linking the fixed costs to facilities with positive outflow are either the strong (disaggregated) constraints (9a) or the weak (aggregated) form (9b). Similarly, two alternative linear programming relaxations, *SRS (Strong Relaxation of SPLP)* and *WRS (Weak Relaxation of SPLP)* respectively, will be considered:

Strong and weak LP-relaxations of SPLP: SRS and WRS

Strong (SRS)	*Weak (WRS)*	*both*
mn inequalities	m inequalities	$\min \sum_i \sum_j c_{ij}x_{ij} + \sum_i f_i y_i$
$y_i - x_{ij} \geq 0$	$n_i y_i - \sum_j x_{ij} \geq 0$	$\sum_i x_{ij} = 1$
(disaggregated)	(aggregated)	$y_i \geq 0,\ x_{ij} \geq 0$

All slack variables for the tighter constraints $y_i - x_{ij} \geq 0$ are upwards bounded by $+1$ while a slack variable corresponding to one of the aggregated inequalities in the more loosely constrained formulation can attain any nonnegative value less than n_i, the number of clients which can be supplied from the ith facility. For $n_i = n$, each aggregated constraint is simply the sum of n disaggregated constraints. A feasible solution to SRS will therefore be feasible for WRS as well while the converse is generally not true. Thus, the adjectives

'strong' and 'weak' have the intuitive meaning that SRS with disaggregated constraints is stronger (closer to the underlying SPLP) than the weaker (less tight) form WRS. On the other hand, SRS has $n(m+1)$ constraints and $m(2n+1)$ variables including slack while these figures reduce to $m+n$ and $m(n+2)$ respectively for WRS.

Disregarding other features such as how to select the next node in the search tree from which to branch, a successful branch and bound algorithm is a lucky compromise between the quality of the bounds and the computational effort involved in their evaluation. For designers of branch and bound algorithms for SPLP, the schism at this stage is essentially reduced to a choice between WRS and SRS (or their duals): WRS with significantly fewer constraints will in general require less computational effort in each bound evaluation while SRS in general will produce tighter bounds. Since either of the two linear programs will have to be solved repeatedly, the substantial difference in the number of constraints led most early researchers to exploit WRS. The remainder of the paper however bears considerable evidence of the prudence of choice of SRS.

6.1. Optimum integral solutions to SRS

It is not true that an optimal solution to SRS automatically will be integral and thus solve SPLP itself. Balinski (1965) noted however that "it might be conjectured" but quoted the following counterexample suggested by Gomory:

Example 2. An SPLP (3,3) is defined by

$$f=\begin{pmatrix}1\\1\\1\end{pmatrix}, \qquad C=\begin{Bmatrix}2 & 1 & 1\\1 & 2 & 1\\1 & 1 & 2\end{Bmatrix}.$$

Minimum cost for establishing any subset of at most two facilities is 5 while a minimum cost of 6 is incurred for a establishing all three facilities. An optimal *fractional* and unique solution (x',y') to SRS is $y'_i=\frac{1}{2}$, all i, $x'_{ij}=\frac{1}{2}$, $i\neq j$ and $x_{ii}=0$, all i, for which the objective function of SRS attains the value $\frac{9}{2}$. □

Is it possible to construct an instance of SPLP(m,n) such that $\min\{m,n\}$ is less than 3 and such that the corresponding SRS has an optimal fractional solution better than any discrete? The following proposition settles the question in the negative

Proposition 5. *For any SPLP with* $\min\{m,n\}<3$ *there exists an optimal solution in integers to the strong LP-relaxation SRS.*

Proof. The proposition is trivially true for $\min\{m,n\}<2$.

$m>2$, $n=2$: For the constraints of SRS written in the form $Ax-b$, it suffices to prove that A is totally unimodular i.e. that every square, nonsingular submatrix of A has a determinant equal to ± 1. Upon addition of slack variables s_{ij} to the disaggregated constraints, the coefficient matrix of SRS is

y_1	$\cdots$	y_m	x_{11}	x_{12}	$\cdots$	x_{m1}	x_{m2}	s_{11}	$\cdots$	s_{m2}	R_1	R_2
			1			1					*	
				1	$\cdots$		1					*
1			−1					−1			*	
1				−1					−1			*
	$\cdots$				$\cdots$				$\cdots$		$\cdots$	$\cdots$
		1				−1				−1	*	
		1					−1			−1		*

Every column contains at most two nonzero entries and every entry is 0, +1, −1. The rows have been partitioned into two sets R_1 and R_2 (members of these sets alternate in the actual case as shown) such that the two nonzero entries in a column with the same sign (here the y-columns) do not appear in the same set R_i of rows. Finally, the two nonzero entries in a column with different signs (here the x-columns) belong to the same set R_i of rows. These are the sufficient conditions due to Heller and Tompkins (1956) for establishing the total unimodularity of the matrix.

$m=2$, $n>2$: In this case the matrix reads:

y_1	y_2	x_{11}	$\cdots$	x_{1n}	x_{21}	$\cdots$	x_{2n}	s_{11}	$\cdots$	s_{2n}
		1	$\cdots$		1	$\cdots$				
				1			1			
1		−1						−1		
⋮			$\cdots$							
1				−1					$\cdots$	
	1				−1					
	⋮					$\cdots$				
	1						−1			−1

or

$$A=\left\{\begin{array}{c|c} & 0 \\ B & \hline \\ & -I_n \end{array}\right\}$$

or A transposed,

$$A^{\mathrm{T}}=\left\{\begin{array}{c|c} & B^{\mathrm{T}} \\ \hline 0 & -I_n \end{array}\right\}.$$

By an argument similar to the one used in the previous case, B^{T} is totally unimodular. It is then easily verified (for details, see e.g. Garfinkel and Nemhauser (1972)) that the same applies for A^{T} and hence for A itself. □

Besides defining a class of SPLP's which is optimally solvable by solving the corresponding instances of SRS, Proposition 5 asserts that no smaller (in terms of mn) counterexample than Gomory's SPLP (3,3) exists.

Nevertheless, counterexamples or not, Balinski's conjecture is *almost* correct. Though vaguely formulated and only supported by empirical evidence, we state it formally to emphasize its indebatable significance:

Conclusion

Since all f_i are assumed nonnegative, any optimal solution (x', y') to SRS will satisfy $y_i' = \max_j x_{ij}'$, all i. If any client's entire demand is supplied by the ith facility, then some x_{ij}' will equal 1 and so will y_i'. Since an SRS related to an instance of SPLP generated at random or with 'real-world' data most often posssesses the Single Assignment Property, split assignments (where clients are served by two or more facilities) will seldomly occur and an optimum solution in integers to SRS results. As will be elaborated upon in Sections 7 and 8, several designers of branch and bound codes for SPLP based on SRS or its dual have noted this fact and realized that in the vast majority of cases no branching at all is required upon the initial solution of SRS.