Discrete Optimization 16 (2015) 76–89

Contents lists available at ScienceDirect

Discrete Optimization

journal homepage: www.elsevier.com/locate/disopt

Local search inequalities

Giuseppe Lancia *, Franca Rinaldi, Paolo Serafini

Dipartimento di Matematica e Informatica - University of Udine, via delle Scienze 206, Udine, Italy

ARTICLE INFO

Article history:
Received 7 October 2014
Received in revised form 14 February 2015
Accepted 19 February 2015

Keywords:
Integer linear programming
Local search
Branch and cut

ABSTRACT

We describe a general method for deriving new inequalities for integer programming formulations of combinatorial optimization problems. The inequalities, motivated by local search algorithms, are valid for all optimal solutions but not necessarily for all feasible solutions. These local search inequalities can help in either pruning the search tree at some nodes or in improving the bound of the LP relaxations.

1. Optimization and local search

One of the most effective ways to solve an NP-hard combinatorial optimization problem is to formulate it as an integer program, which is in turn solved by branch and bound [1]. Let the formulation be

$$z_{IP} := \min\{c\,x : x \in S\} \tag{1}$$

where $S = \{x : A\,x \geq b, x \geq 0, x \in \mathbb{Z}^n\}$. Moreover, let $P = \{x : A\,x \geq b, x \geq 0\}$ and $P^I = \text{conv}(S)$.

Valid inequalities (also called cuts) can be added to (1) to strengthen the quality of the LP bound to be used in the branch-and-bound. The process of solving a combinatorial optimization problem with the addition of valid cuts is called *branch-and-cut* [2].

Local search is a general framework for finding good (not necessarily optimal) solutions of an optimization problem [3,4]. In local search, a neighborhood function $\mathcal{N}(s)$ is specified, which, for a feasible solution s, defines a set of feasible solutions "close" to s. A *local optimum* is a solution s^* such that $c\,s^* \leq c\,s$ for all $s \in \mathcal{N}(s^*)$. Clearly, an optimal solution of the problem is also a local optimum for each possible neighborhood but not vice-versa. Local search heuristics usually work by quickly finding as many local optima as possible, and then returning the best one.

The results presented in this paper are based on the following observation:

> *Given a local search neighborhood $\mathcal{N}$, a global optimum of the problem must also be a local optimum for $\mathcal{N}$. This is therefore an additional constraint on the global optimum.*

If it is possible to express the above constraint via linear inequalities, we call each such linear constraint a *local search inequality* (LSI). Basically, local search inequalities are constraints saying that, for each move which changes a feasible solution x into a feasible solution $x' \in \mathcal{N}(x)$, it must be $c\,x' \geq c\,x$. These constraints are valid for local optima (and hence for global optima), but may be violated by some other feasible solutions in P^I. Therefore, they cut *through* the set P^I and are not valid inequalities in the usual sense.

We mention that the idea of using inequalities that are not valid for all feasible solutions but only for some subsets of solutions can be also found in other settings. As examples of such inequalities, we cite the *logic cuts* which can be found in [5,6] and the *conditional inequalities* appearing in [7].

* Corresponding author.
E-mail addresses: giuseppe.lancia@uniud.it (G. Lancia), franca.rinaldi@uniud.it (F. Rinaldi), paolo.serafini@uniud.it (P. Serafini).

http://dx.doi.org/10.1016/j.disopt.2015.02.003
1572-5286/

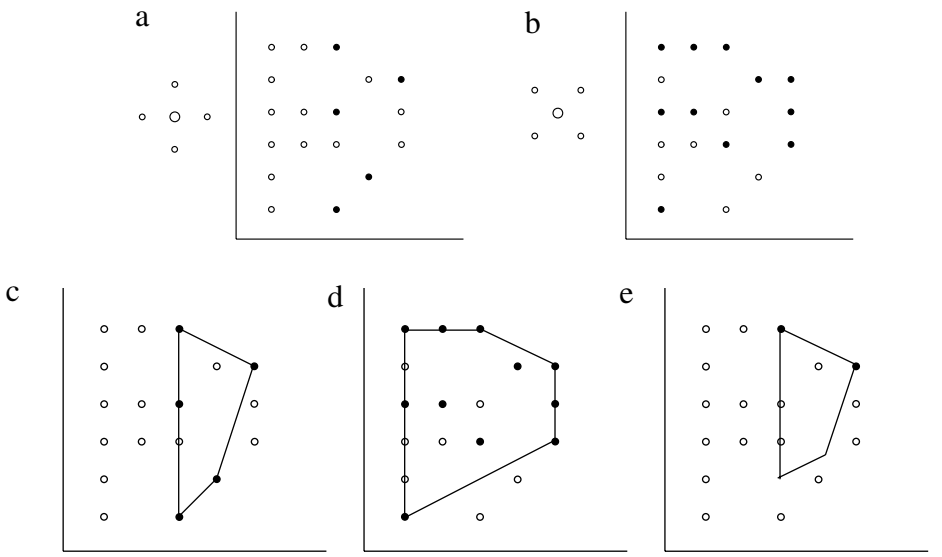


Fig. 1. A feasible set with two neighborhood functions $\mathcal{N}^1$ (a) and $\mathcal{N}^2$ (b). In (a) neighbor points are obtained by moving vertically or horizontally; in (b) by moving diagonally. The objective function is $\max x_1 + x_2$ and the local optima are the black dots. $P^{\mathcal{N}_1}$ is shown in (c) and $P^{\mathcal{N}_2}$ is shown in (d). $P^{\mathcal{N}_1} \cap P^{\mathcal{N}_2}$ is shown in (e) which has only two local optima points, one of which is the global optimum.

Given a neighborhood function $\mathcal{N}$, let $X(\mathcal{N})$ be the set of incidence vectors of all local optima for $\mathcal{N}$, and let $P^{\mathcal{N}} = \text{conv}(X(\mathcal{N}))$. Then, a LSI is nothing but a valid inequality for $P^{\mathcal{N}}$. The polytope $P^{\mathcal{N}}$ is contained in P^I, and if k distinct neighborhood functions are considered, the optimal solutions in P^I are contained in $P^{\mathcal{N}_1} \cap P^{\mathcal{N}_2} \cap \cdots \cap P^{\mathcal{N}_k}$. In Figs. 1(a) and (b) a feasible set is shown with two neighborhood functions $\mathcal{N}^1$ and $\mathcal{N}^2$. The neighborhood functions are schematically shown at the left of the figures (in the first case neighbor points are obtained moving vertically or horizontally; in the second case moving diagonally). If the objective function is $\max x_1 + x_2$ then the local optima are the black dots. In Figs. 1(c) and (d) we can see $P^{\mathcal{N}_1}$ and $P^{\mathcal{N}_2}$. In Fig. 1(e) we see $P^{\mathcal{N}_1} \cap P^{\mathcal{N}_2}$ which has only two local optima points, one of which is the global optimum.

From our computational experiments, we have noticed that $X(\mathcal{N}_1) \cap X(\mathcal{N}_2) \cap \cdots \cap X(\mathcal{N}_k)$ is typically much smaller than S, even for $k = 2, 3$. For example, consider the problem of finding the optimal TSP tour over the first 13 nodes of the TSPLIB instance `fri26`. We computed the local optima for three simple neighborhood functions defined by the moves that exchange two consecutive cities along the tour ($\mathcal{N}_1$), exchange any two cities along the tour ($\mathcal{N}_2$) and remove two edges from the tour and reconnect the two resulting paths ($\mathcal{N}_3$). Then, among the 239,500,800 tours, there are 432,507 local optima for $\mathcal{N}_1$, 7,293 local optima for $\mathcal{N}_2$ and 3 local optima for $\mathcal{N}_3$. Moreover, $X(\mathcal{N}_1) \cap X(\mathcal{N}_2) \cap X(\mathcal{N}_3)$ has just 2 elements, and they are both global optima.

Local search inequalities are, by their nature, very general, since they apply whenever a local optimality condition can be expressed by linear inequalities. In this paper, we will focus on the Traveling Salesman (TSP), on the Maximum Cut (Max-Cut) and on the Maximum Satisfiability problems (Max-SAT).

In a sense, LSIs are "formulation-independent", i.e., they can be added to any formulation whose variables include the variables appearing in the LSIs. For instance, there are several distinct formulations for the TSP which have, among their variables, binary variables x_{ij} associated to the edges of the graph (see [8]). Then, if we have a set of LSIs involving only the edge variables, they could be added to any of these formulations.

Another nice property of LSIs is that, as long as the neighborhood function is relatively simple (such as exchanging the order of two elements in a permutation), the number of corresponding inequalities is fairly small, and LSIs can be directly added to the formulation without the need of a separation algorithm. We will see examples of this type later on. On the other hand, for more complex neighborhood functions, it may be the case that there is an exponential number of LSIs, but still they can be dealt with in polynomial time via a separation algorithm. We will see examples of this type as well.

The remainder of the paper is organized as follows. In Section 2 we describe two neighborhoods for the Symmetric TSP and some corresponding LSIs. In Section 3 and Section 4 we describe LSIs for the Max-Cut and the Max-SAT problems, respectively. In Section 5 we report the results of some computational experiments. Some conclusions are drawn in Section 6.

Notation. We will adopt the following notation. For a graph $G = (V, E)$ and a set of nodes $S \subset V$, we denote by $\delta(S)$ the set of edges with one endpoint in S and the other in $V \setminus S$. With a slight abuse of notation, we write $\delta(v)$ instead of $\delta(\{v\})$ when $|S| = 1$. By $N(v)$ we denote the set of neighbors of $v \in V$, i.e. $N(v) := \{u : uv \in E\}$. If x are variables of a linear program with indices in a set I, and $J \subseteq I$, by $x(J)$ we denote the sum $\sum_{i \in J} x_i$.

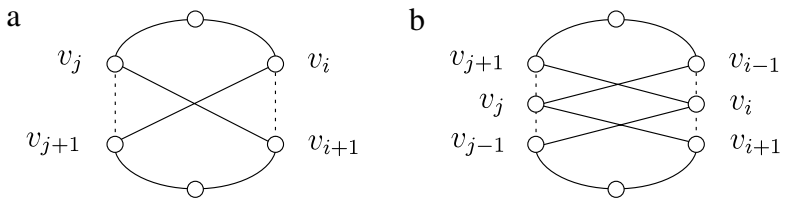


Fig. 2. 2-OPT and SWAP moves.

2. The Traveling salesman problem

The (symmetric) Traveling salesman problem (TSP) is the following problem: Given an undirected graph $G = (V, E)$, with costs c_e on the edges $e \in E$, find a hamiltonian tour τ minimizing $c(\tau) := \sum_{e\in\tau} c_e$. Without loss of generality, we can assume G is the complete graph K_n and $V = \{1, \ldots, n\}$.

The TSP is a widely known problem on which almost all combinatorial optimization approaches have been tried [9]. Almost all ILP formulations for the TSP employ, among others, binary variables x_e, associated to the edges $e \in E$, to select the edges of the tour (for a survey about TSP formulations see [10]).

A popular and very effective formulation consists in minimizing $\sum_{e\in E} c_e x_e$ subject to the *degree constraints* (2) and the *subtour elimination constraints* (3):

$$x(\delta(v)) = 2 \quad v \in V \tag{2}$$

$$x(\delta(S)) \geq 2 \quad S \subseteq V : 2 \leq |S| \leq n - 2. \tag{3}$$

We have used this formulation in our computational experiments to test the effectiveness of LSIs. Several local search neighborhoods have been proposed for the TSP [11]. In this section we consider the following two:

2-OPT Neighborhood: Given a tour $H = (v_1, v_2, \ldots, v_n, v_1)$, remove two edges and replace them with two new ones, to obtain a new tour. For instance, replacing the edges $v_i v_{i+1}$ and $v_j v_{j+1}$ in H yields the new tour

$$H' = (v_1, \ldots, v_i, v_j, v_{j-1}, \ldots, v_{i+1}, v_{j+1}, \ldots, v_1)$$

(see Fig. 2(a)).

(Node) SWAP Neighborhood: Given a tour $H = (v_1, v_2, \ldots, v_n, v_1)$, pick any two vertices and exchange them within the tour. For instance, swapping the vertices v_i and v_j in H yields the new tour

$$H' = (v_1, \ldots, v_{i-1}, v_j, v_{i+1}, \ldots, v_{j-1}, v_i, v_{j+1}, \ldots, v_1)$$

(see Fig. 2(b)).

The 2-OPT neighborhood can be generalized to k-OPT, in which k edges are removed and replaced by some other edges yielding a new tour. It can be observed that the SWAP move is in fact a special type of 4-OPT move, in which four edges are removed and replaced by four new ones.

We will now present LSIs based on the above neighborhoods. The inequalities of the first family, called *2-OPT inequalities*, are LSIs based on the fact that an optimal tour cannot be improved by a 2-OPT move. Similarly, the *SWAP inequalities* state that an optimal tour cannot be improved by a SWAP move.

2-OPT inequalities

Denote by P^{2O} the convex hull of the incidence vectors of the local optimal solutions for the 2-OPT neighborhood. In this section we introduce some inequalities that are valid for P^{2O}.

Given any subset S of V with four nodes (without loss of generality we hereafter assume $S = \{1, 2, 3, 4\}$), we partition $E(S) := \{ij \in E : i, j \in S\}$ in three disjoint pairs of edges, i.e., the *horizontal* edges α, the *crossing* edges β and the *vertical* edges γ (see Fig. 3). Without loss of generality we assume $c(\alpha) \geq c(\beta) \geq c(\gamma)$ where $c(p)$ is the cost of pair p for each pair $p = \alpha, \beta, \gamma$. Given the labeling of the nodes as in Fig. 3, we have $\alpha = \{12, 34\}$, $\beta = \{14, 23\}$ and $\gamma = \{13, 24\}$. In the following we denote by $x(\alpha) := x_{12} + x_{34}$, $x(\beta) := x_{14} + x_{23}$ and $x(\gamma) := x_{13} + x_{24}$.

Proposition 1. *Assume*

$$c(\alpha) > c(\beta) \geq c(\gamma).$$

Then the inequality

$$x(\alpha) \leq 1 \tag{4}$$

is valid for P^{2O}.

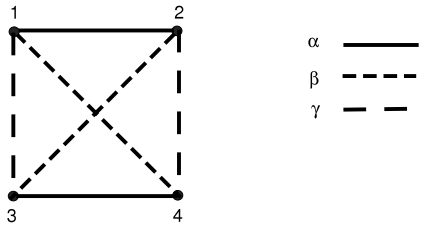


Fig. 3. Partition of $E(\{1, 2, 3, 4\})$ in α, β and γ.

Proof. Let $\bar{x}$ be the incidence vector of a feasible tour τ. Then, if $\bar{x}(\alpha) = 2$, the two edges in α could be removed from τ and replaced by either the edges in β or those in γ to obtain a better tour. Hence, τ would not be a local optimum. ■

Proposition 2. *Assume*

$$c(\alpha) \geq c(\beta) > c(\gamma).$$

Then the inequality

$$x(\alpha) + x(\beta) \leq 2 \tag{5}$$

is valid for P^{2O}.

Proof. Let $\bar{x}$ be the incidence vector of a feasible tour τ. If $x(\bar{\alpha}) + x(\bar{\beta}) \geq 3$, then τ either contains both α edges and an edge of β, or it contains both β edges and an edge of α. In the first case, by replacing the edges α with γ we would obtain a better tour. Similarly, in the second case, we could obtain a better tour by replacing β with γ. ■

From the degree constraints (2) and the subtour elimination inequalities (3) it follows that any incidence vector of a tour satisfies the condition

$$(x(\alpha) = 0) \vee (x(\beta) = 0) \vee (x(\gamma) = 0)\,. \tag{6}$$

Proposition 3. *Assume*

$$c(\alpha) > c(\beta) > c(\gamma).$$

Then the following inequalities are valid for P^{2O}*:*

$$2\,x(\alpha) + x(\beta) + x_{e^\gamma} \leq 3 \qquad \text{for each } e^\gamma \in \gamma \tag{7}$$

$$2\,x(\alpha) + 2\,x_{e_1^\beta} + x_{e_2^\beta} + x(\gamma) \leq 4 \quad \text{for each } e_1^\beta, e_2^\beta \in \beta,\ e_1^\beta \neq e_2^\beta \tag{8}$$

$$3\,x(\alpha) + 2\,x(\beta) + x(\gamma) \leq 5. \tag{9}$$

Proof. Let $\bar{x}$ be the incidence vector of a 2-OPT local optimum tour τ. By (6) $\bar{x}(\alpha) = 0$ or $\bar{x}(\beta) = 0$ or $\bar{x}(\gamma) = 0$. In the first case, all the inequalities are satisfied because of the subtour elimination constraint $\bar{x}(\alpha) + \bar{x}(\beta) + \bar{x}(\gamma) \leq 3$ induced by the set $S = \{1, 2, 3, 4\}$. In the second case, they follow from Proposition 1. In the third case, they follow from Propositions 1 and 2. ■

We notice that for each inequality of type (7)–(9) one can find a feasible fractional solution that satisfies that inequality as an equality while satisfies each other inequality and the subtour elimination constraints strictly.

We have also derived the inequalities (7)–(9) directly by computing all facets of the projection of the polytope P^{2O}. The inequalities (7)–(9) are indeed facet defining. There are other facet defining inequalities but they are less effective. Considering the set of six arcs $\alpha \cup \beta \cup \gamma$, there are 27 incidence vectors of subsets belonging to tours, and 7 of them are not local optima (assuming the ordering in Proposition 3). The inequalities (7) and (9) cut all non local optima and the inequalities (8) cut all but one. The other inequalities (not listed here) cut less non local optima. In particular one cuts none of the non local optima. It turns out that it is a subtour inequality.

The next proposition introduces a new family of inequalities with an exponential number of elements.

Proposition 4. *Assume*

$$c(\alpha) \geq c(\beta) > c(\gamma).$$

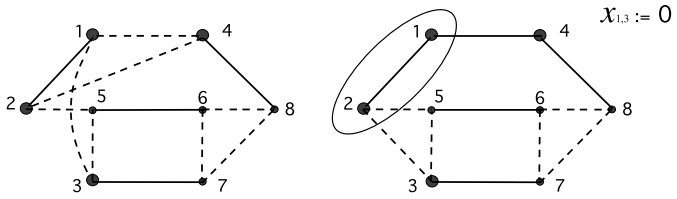
Then for each subset T *of nodes such that* $1, 2 \in T$ *and* $3, 4 \notin T$ *the inequality*

$$x(\delta(T)) \geq 2\,x(\beta) \tag{10}$$

is valid for P^{2O}.

Table 1
The TSP instance of the example.

	1	2	3	4	5	6	7	8
1	–	100	159	102	146	165	219	670
2	100	–	68	8	40	61	135	62
3	159	68	–	76	55	74	62	116
4	102	8	76	–	45	64	134	27
5	146	40	55	45	–	25	111	63
6	165	61	74	64	25	–	88	59
7	219	135	62	134	111	88	–	86
8	670	62	116	27	63	59	86	–



(a) Root solution. (b) Subproblem solution.

Fig. 4. LP solutions for the example.

Proof. Let $\bar{x}$ be the incidence vector of a 2-OPT local optimum tour τ. If $\bar{x}(\beta) \leq 1$, then inequality (10) is dominated or equal to the subtour elimination constraint induced by T. Assume $\bar{x}(\beta) = 2$. Since, despite the fact that $c(\gamma) < c(\beta)$, the local optimal tour τ uses both edges in β, then τ must contain a path p' between 1 and 3 and a path p'' between 3 and 4. Therefore, the cut $\delta(T)$ contains, besides the edges of β, at least one edge of p' and at least one edge of p'', thus at least $4 = 2\,\bar{x}(\beta)$ edges of τ. ■

Separating 2-OPT LSI. Each set of four nodes identifies one inequality of type (4), (5) and (7)–(9) and an exponential number of inequalities of type (10). As a consequence, the $O(n^4)$ inequalities (4), (5) and (7)–(9) can be either added to the initial model or separated in polynomial time by a direct check. On the other hand, the separation of inequalities (10) requires the adoption of a cutting plane approach. For each set of four nodes, they can always be labeled as 1, 2, 3, 4 in such a way that $c(\alpha) \geq c(\beta) \geq c(\gamma)$. If $c(\beta) > c(\gamma)$ the separation problem of (10) with respect to these nodes is the following.

Separation problem for inequalities (10): Given $\hat{x} \in \mathbb{R}^{|E|}$ and nodes 1, 2, 3, 4 for which condition

$$c_{12} + c_{34} \geq c_{14} + c_{23} > c_{13} + c_{24}$$

holds, find, if it exists, a subset $T \subseteq V$, with $1, 2 \in T$, $3, 4 \notin T$ such that $\sum_{e \in \delta(T)} \hat{x}_e < 2\,(\hat{x}_{14} + \hat{x}_{23})$.

The above separation problem can be solved in polynomial time as a maximum flow problem as follows. Define a directed graph $\bar{G} = (\bar{V}, \bar{E})$ with $n - 2$ nodes, where two special nodes s and t are obtained by contraction of $\{1, 2\}$ and $\{3, 4\}$, respectively, and the remaining $n - 4$ nodes correspond to the other nodes of G. For each $ij \in E$ with $i, j \notin \{1, 2, 3, 4\}$, the arcset $\bar{E}$ contains both arcs (i, j) and (j, i) with capacity $\hat{x}_{ij}$. Moreover, $\bar{E}$ contains an arc (s, i), for each $i \neq 3, 4$, with capacity $\hat{x}_{1i} + \hat{x}_{2i}$, an arc (j, t), for each $j \neq 1, 2$, with capacity $\hat{x}_{3j} + \hat{x}_{4j}$ and the arc (s, t) with capacity $\hat{x}_{13} + \hat{x}_{14} + \hat{x}_{23} + \hat{x}_{24}$ (note that arcs with zero capacity can in fact be omitted from $\bar{E}$). Clearly, each subset $\bar{T}$ of nodes of $\bar{G}$ containing s and not t corresponds to a subset T of nodes of G with $1, 2 \in T$ and $3, 4 \notin T$. Moreover, the capacity of the cut $\delta(\bar{T})$ in $\bar{G}$ is equal to the left hand side of inequality (10) evaluated in $\hat{x}$. Therefore inequalities (10) are satisfied by $\hat{x}$ if and only if the minimum cut separating s from t in $\bar{G}$ has capacity at least $2\,\hat{x}(\beta)$. By the max flow-min cut theorem, a cut of minimum capacity separating s and t in $\bar{G}$ may be found by solving a maximum flow problem.

We remark that when $\bar{x}(\beta) \leq 1$ the inequalities (10) are dominated by the subtour elimination constraints. We note the interesting fact that there may be solutions that do not violate subtour inequalities but violate both comb inequalities [12] and local search inequalities. These solutions can be cut off by local search inequalities without the need of resorting to separation routines for comb inequalities. In the following example we show an occurrence of this fact.

An example. We now describe the effect of the 2-OPT local search inequalities when added to the subtour elimination model of the TSP. Consider the instance over 8 nodes reported in Table 1. The root node solution x^* is shown in Fig. 4(a) where a solid line represents an edge of value 1 and a dashed line represents an edge of value 0.5. The value of x^* is 512.5 and the solution does not violate any 2-OPT local search inequality. After branching on the edge 13, we consider the subproblem in which $x_{13} = 0$. The optimal solution $\hat{x}$ of the subtour elimination relaxation is shown in Fig. 4(b) and has value 514. This solution satisfies all subtour inequalities but violates the comb inequality given by the handle $\{2, 3, 5\}$ and the teeth $\{1, 2\}$, $\{5, 6\}$ and $\{3, 7\}$, i.e.

$$x_{23} + x_{25} + x_{35} + x_{12} + x_{56} + x_{37} \leq 4.$$

Moreover, $\hat{x}$ violates local search inequalities of types (5) and (7), (8) and (9) corresponding to the four nodes 1, 2, 3, 4. In this case $c(\alpha) = c_{12} + c_{34} = 176$, $c(\beta) = c_{14} + c_{23} = 170$ and $c(\gamma) = c_{13} + c_{24} = 167$. We have

$$\hat{x}(\alpha) + \hat{x}(\beta) = 1 + 1.5 = 2.5 > 2,$$
$$2\hat{x}(\alpha) + \hat{x}(\beta) + \hat{x}_{24} = 2 + 1.5 + 0 = 3.5 > 3$$
$$2\hat{x}(\alpha) + 2\hat{x}_{14} + \hat{x}_{23} + \hat{x}(\gamma) = 2 + 2 + 0.5 + 0 = 4.5 > 4,$$
$$3\hat{x}(\alpha) + 2\hat{x}(\beta) + \hat{x}(\gamma) = 3 + 3 + 0 = 6 > 5.$$

Moreover, $\hat{x}$ violates the inequality of type (10) corresponding to the set $T = \{1, 2\}$ separating $\{1, 2\}$ from $\{3, 4\}$. Indeed $\hat{x}(\beta) = 1.5$ while the cut $\delta(T)$ has capacity $2 < 2 \times 1.5 = 3$. We observed that after the addition of any of the corresponding violated inequalities the new solution is the global optimum tour (1, 4, 8, 7, 3, 6, 5, 2, 1) of value 516.

SWAP inequalities

Let us consider the SWAP neighborhood, in which a new tour τ' is obtained from a given one τ by exchanging any two nodes i and j in τ (see Fig. 2 with $i := v_i$ and $j := v_j$).

Proposition 5. *All local optima with respect to the SWAP neighborhood satisfy the following LSIs:*

$$\sum_{k \neq j} c_{ik}x_{ik} + \sum_{k \neq i} c_{jk}x_{jk} \leq \sum_{k \neq i} c_{ik}x_{jk} + \sum_{k \neq j} c_{jk}x_{ik} \quad i, j \in V. \tag{11}$$

Proof. Suppose that in a local optimum tour x the nodes i and j are not adjacent. Hence the value of the lhs is the value of the two edges incident to i plus the value of the two edges incident to j in the tour x. This is the value of the edges that the SWAP move would delete from the tour. Now note that $c_{ik}x_{jk}$ represents the cost of going from k to i when k is adjacent to j in the tour. In other words, the tour passes through the edge jk, but we exploit this fact to pick up the cost of the edge ik, which will be brought into the tour by the SWAP move. Hence the rhs is the value of the new edges. Then the inequality simply states the local optimality of x. If i and j are adjacent in the tour the inequality is still valid. The edge ij, which is present also after the SWAP move, does not enter (11) and the lhs and the rhs amount to the cost of two edges, instead of four. ■

While in the inequalities (11) the coefficients of the variables are given by the edge costs, in the 2-OPT LSIs the variables have low-value constant coefficients (either 1, 2 or 3). Usually, having low coefficients is preferable, as in LP relaxations variables with high coefficients tend to assume small fractional values. We then propose the following "combinatorial" version of the SWAP inequalities in which all variables have coefficient 1.

Proposition 6. *For each choice of six nodes i', i, i'', j', j, j'' such that*

$$c_{i'i} + c_{ii''} + c_{j'j} + c_{jj''} > c_{j'i} + c_{ij''} + c_{i'j} + c_{ji''}$$

the following LSI must be satisfied by each SWAP local optimum:

$$x_{i'i} + x_{ii''} + x_{j'j} + x_{jj''} \leq 3. \tag{12}$$

Proof. If all four variables were equal to 1, it would mean that the tour visits i', i, i'' in sequence and j', j, j'' in sequence. By swapping i and j we would then obtain a better tour. ■

Both inequalities (11) and (12) can be separated by complete enumeration in polynomial time (or directly added to the model) as there are $O(n^2)$ inequalities (11) and $O(n^6)$ inequalities (12).

3. The Max-Cut problem

Given an undirected graph $G = (V, E)$, the cardinality Max-Cut problem requires finding a subset $S \subset V$ such that $|\delta(S)|$ is maximum. The problem is strongly NP-hard [13]. A possible neighborhood for Max-Cut is the following:

> k-**FLIP Neighborhood**: Given a solution $S \subset V$, pick a subset A of k nodes and "flip their color" (i.e., if a node is in S move it to $V \setminus S$, while if it is $V \setminus S$ move it to S).

We note that a k-flip on a subset A of nodes changes the state only of edges in $\delta(A)$. In particular, an edge $e \in \delta(A)$ belongs to the cut after the move if and only if it was not in the cut before.

Consider any formulation of the problem which, possibly among others, has variables x_{ij} for each pair i, j in V such that $x_{ij} = 1$ iff $|S \cap \{i, j\}| = 1$. Note that a subset of these variables correspond to the edges of E. It is immediate to verify that each local optimum has to satisfy the following local search inequalities:

$$\sum_{e \in \delta(A)} x_e \geq \left\lceil \frac{|\delta(A)|}{2} \right\rceil \quad A \subset V. \tag{13}$$

In particular, the case $|A| = 1$ is the most commonly used in local search procedures for Max-Cut, and the corresponding 1-FLIP LSIs are

$$\sum_{e\in\delta(i)} x_e \geq \left\lceil \frac{|\delta(i)|}{2} \right\rceil \quad i \in V. \tag{14}$$

For $|A| = 2$, (13) become the following 2-FLIP LSIs

$$x(\delta(i)) + x(\delta(j)) - 2\,[ij \in E]\,x_{ij} \geq \left\lceil \frac{|\delta(\{i,j\})|}{2} \right\rceil \quad i \in V,\ j \in V,\ i \neq j \tag{15}$$

where $[ij \in E] = 1$ if $ij \in E$ and 0 otherwise.

From (15) one can derive other, possibly stronger, families of local cuts by separately considering the cases $x_{ij} = 0$ and $x_{ij} = 1$. We call these LSIs the I2-FLIP (standing for "Improved 2-FLIP"). For each pair of vertices $a, b \in V$, we define $N(a-b) := N(a) \setminus (N(b) \cup \{b\})$ and $N(a+b) := (N(a) \cup N(b)) \setminus \{a, b\}$. In other words $N(a-b)$ is the set of vertices, different from b, that are adjacent to a but not to b and $N(a+b)$ is the set of vertices that are adjacent to the subset $\{a, b\}$.

Let us consider the case that, in a feasible solution, $x_{ij} = 0$. Then i and j are on the same shore of the cut and $x_{ik} = x_{jk}$ for each $k \neq i, j$. The constraints (15) can be used to derive a family of inequalities, each of which is of the general form

$$\sum_{k\in N(i+j)} |N(k) \cap \{i,j\}|\, x_{ak} \geq \left\lceil \frac{|\delta(\{i,j\})|}{2} \right\rceil (1 - x_{ij}) \tag{16}$$

where, for each $k \in N(i+j)$, a can be any index in the set $\{i, j\}$. Note that the left hand side of (16) corresponds to the sum $x(\delta(i)) + x(\delta(j))$ since for each $k \in N(i) \cap N(j)$ it is $x_{ik} + x_{jk} = 2\,x_{ak} = |N(k) \cap \{i,j\}| x_{ak}$ for each $a = i, j$. The inequalities (16) are trivially satisfied when $x_{ij} = 1$.

Since for each pair i, j and $k \in N(k) \cap \{i, j\}$ the variable x_{ak} could be either x_{ik} or x_{jk}, there are an exponential number of inequalities (16). These inequalities have a simple separation algorithm. Namely, given a fractional solution $\bar{x}$, to obtain the most violated inequality for a pair i, j, we set $a = i$ if $\bar{x}_{ik} \leq \bar{x}_{jk}$ and $a = j$ otherwise.

Let us now consider the case that, in a feasible solution, $x_{ij} = 1$. Then i and j are on different shores and each vertex $k \in N(i) \cap N(j)$ contributes exactly one edge to the cut (either ik or jk), and this remains true also after the 2-FLIP. However, for all the remaining edges in $\delta(\{i, j\})$ an edge is in the new cut if and only if it was not in the cut before. Therefore, at a local optimum at least half of them are in the cut. This can be enforced via the following constraints:

$$\sum_{k\in N(i-j)} x_{ik} + \sum_{k\in N(j-i)} x_{jk} \geq \left\lceil \frac{|N(i) \ominus N(j)|}{2} \right\rceil x_{ij} \tag{17}$$

where the symbol $\ominus$ denotes the symmetric difference of two sets. Note that all the inequalities (17) are trivially satisfied when $x_{ij} = 0$.

The literature for Max-Cut includes integer programming [14–17] and semidefinite programming [18] approaches. In general, semidefinite programming is the best approach for dense graphs, while branch-and-cut is usually the most effective procedure when the graphs are very sparse. In our experiments we used the following well-known *triangle inequalities formulation* [14]. The (binary) variables of this formulation are x_{ij} for each pair of vertices i, j, while the constraints are

$$x_{jk} - x_{ij} - x_{ik} \leq 0 \tag{18}$$

$$x_{ik} - x_{ij} - x_{jk} \leq 0 \tag{19}$$

$$x_{ij} - x_{ik} - x_{jk} \leq 0 \tag{20}$$

$$x_{ij} + x_{jk} + x_{ik} \leq 2 \tag{21}$$

for all triples of vertices i, j, k.

4. The Max-SAT problem

Given n Boolean variables $v_1, \ldots, v_n$, a *literal* is either a variable v_i or its negation $\bar{v}_i$. A *clause* is the disjunction of a set of literals, and a Boolean formula, in conjunctive form, is the conjunction of a set of clauses. A clause is *satisfied* by a truth assignment of the variables if at least one of its literals is true. The Maximum Satisfiability (Max-SAT) problem requires to find an assignment of truth values for all the variables which satisfies the maximum number of clauses in a given Boolean formula.

With a slight abuse of terminology, we will identify a clause with its underlying set of literals, so that we will write $l \in C$ meaning that the literal l appears in the clause C.

Given a Boolean formula with variables $v_1, \ldots, v_n$ and clauses $C_1, \ldots, C_m$, for each variable v_i we denote by $C^+(v_i) := \{j : v_i \in C_j\}$ the set of indices of the clauses containing the literal v_i, by $C^-(v_i) := \{j : \bar{v}_i \in C_j\}$ the set of indices of the clauses containing the literal $\bar{v}_i$ and define $C(v_i) := C^+(v_i) \cup C^-(v_i)$. Similarly, we denote by $V^+(C_j) := \{i : v_i \in C_j\}$ and

by $V^-(C_j) := \{i : \bar{v}_i \in C_j\}$ the sets of indices of the variables whose literal v_i and, respectively, $\bar{v}_i$ appears in C_j and define $V(C_j) := V^+(C_j) \cup V^-(C_j)$.

The Max-SAT problem can be formulated as the following ILP problem with 0–1 variables $x_1, \ldots, x_n$ associated to the n Boolean variables and 0–1 variables $y_1, \ldots, y_m$ associated to the m clauses. The meaning of the x variables is that $x_i = 1$ when v_i is set to TRUE, while $y_j = 1$ when clause C_j is satisfied.

$$\begin{aligned} \max \quad & \sum_j y_j \\ & y_j \le \sum_{i \in V^+(C_j)} x_i + \sum_{i \in V^-(C_j)} (1 - x_i) \quad j = 1, \ldots, m \\ & x_i, y_j \in \{0, 1\} \quad i = 1, \ldots, n,\ j = 1, \ldots, m. \end{aligned} \tag{22}$$

We consider the following neighborhood for Max-SAT:

1-FLIP Neighborhood: Given a truth assignment for all the variables, pick a variable v_i and change its truth value.

It is immediate to verify that each local optimum has to satisfy the following LSIs, called 1-FLIP inequalities:

$$\sum_{j \in C(v_i)} y_j \ge \left\lceil \frac{|C(v_i)|}{2} \right\rceil \quad i = 1, \ldots, n. \tag{23}$$

Indeed, if this were not the case for a variable v_i, by swapping the truth value of v_i we would obtain a better solution. This idea can be strengthened by observing that the best assignment for the variable v_i can guarantee to satisfy the largest between the set of clauses $C^+(v_i)$ and $C^-(v_i)$. Therefore, we obtain the following LSI:

$$\sum_{j \in C(v_i)} y_j \ge \max\{|C^+(v_i)|, |C^-(v_i)|\} \quad i = 1, \ldots, n.$$

If we are willing to add more variables, then these LSIs can be further strengthened as follows. For each variable v_i and each clause $C_j, j \in C(v_i)$, we introduce a new 0–1 variable z_{ij} such that $z_{ij} = 1$ if and only if C_j is satisfied by a variable different from v_i (irrespective of v_i satisfying C_j). Hence,

$$z_{ij} = \max \left\{ \max_{r \in V^+(C_j) \setminus \{i\}} x_r, \max_{r \in V^-(C_j) \setminus \{i\}} 1 - x_r \right\},$$

which leads to the $|C_j|$ linear inequalities:

$$\begin{aligned} & z_{ij} \ge x_r && r \in V^+(C_j) \setminus \{i\} \\ & z_{ij} \ge 1 - x_r && r \in V^-(C_j) \setminus \{i\} \\ & z_{ij} \le \sum_{r \in V^+(C_j) \setminus \{i\}} x_r + \sum_{r \in V^-(C_j) \setminus \{i\}} (1 - x_r). \end{aligned} \tag{24}$$

Hence the following LSIs are valid for the 1-FLIP local optima:

$$\sum_{j \in C(v_i)} y_j \ge |C^+(v_i)| + \sum_{j \in C^-(v_i)} z_{ij} \quad \forall\, i = 1, \ldots, n \tag{25}$$

$$\sum_{j \in C(v_i)} y_j \ge |C^-(v_i)| + \sum_{j \in C^+(v_i)} z_{ij} \quad \forall\, i = 1, \ldots, n. \tag{26}$$

Indeed, given a truth assignment, by setting $v_i =$ TRUE and keeping all the other variables at their values, we obtain a truth assignment for which all clauses in $C^+(v_i)$ are satisfied while all the clauses in $C^-(v_i)$ which are satisfied by variables different from v_i remain satisfied. This explains the inequalities (25). A similar argument applies to setting $v_i =$ FALSE for the inequalities (26). We call the inequalities (25) and (26) I1-FLIP, for Improved 1-FLIP. They require to add $\sum_j |V(C_j)|$ variables and $\sum_j |V(C_j)|$ constraints to the original model.

Instead of introducing the z variables and the constraints (25) and (26), one can use an equivalent formulation with an exponential number of constraints that allows for an easy separation algorithm. Note that, by (24), we can replace the variable z_{ij} in (25) with either x_r, for any $r \in V^+(C_j) \setminus \{i\}$, or $1 - x_r$, for any $r \in V^-(C_j) \setminus \{i\}$, and still obtain a valid inequality. By doing this for all clauses $j \in C^-(v_i)$ we obtain a family of valid inequalities of the form, for $i = 1, \ldots, n$,

$$\sum_{j \in C(v_i)} y_j \ge |C^+(v_i)| + \sum_{j \in C^-(v_i)} \Delta\left(r(j), V^+(C_j)\right) x_{r(j)} + \sum_{j \in C^-(v_i)} \Delta\left(r(j), V^-(C_j) \setminus \{i\}\right) (1 - x_{r(j)}) \tag{27}$$

where, for each $j \in C^-(v_i)$, $r(j)$ can be any index in $V(C_j) \setminus \{i\}$ and $\Delta(a, A)$ denotes a function which, for an element a and a set A, is 1 if $a \in A$ and 0 otherwise. Furthermore, for each solution $(\bar{x}, \bar{y}, \bar{z})$, each variable-index i and each $j \in C^-(v_i)$, there is at least a choice of the index $r(j) \in V(C_j) \setminus \{i\}$ such that

$$\bar{z}_{ij} = \Delta\left(r(j), V^+(C_j)\right) \bar{x}_{r(j)} + \Delta\left(r(j), V^-(C_j) \setminus \{i\}\right) (1 - \bar{x}_{r(j)})$$

holds.

Similarly, for each $i = 1, \ldots, n$ we can replace constraints (26) with the equivalent family of constraints

$$\sum_{j \in C(v_i)} y_j \geq |C^-(v_i)| + \sum_{j \in C^+(v_i)} \Delta\left(r(j), V^+(C_j) \setminus \{i\}\right) x_{r(j)} + \sum_{j \in C^+(v_i)} \Delta\left(r(j), V^-(C_j)\right) (1 - x_{r(j)}) \tag{28}$$

containing an element for each choice of indices $r(j), j \in C^+(v_i)$, in the sets $V(C_j) \setminus \{i\}$.

We remark that, although there is an exponential number $O(n^m)$ of inequalities (27) and (28), these inequalities allow for a trivial separation algorithm. Indeed, given a solution $(\bar{x}, \bar{y})$ of the LP relaxation of (22), for each variable-index i and for each $j \in C(v_i)$ we can easily compute an index $\bar{r}(j)$ that achieves the maximum

$$\max \left\{ \max_{r \in V^+(C_j) \setminus \{i\}} \bar{x}_r, \max_{r \in V^-(C_j) \setminus \{i\}} 1 - \bar{x}_r \right\}.$$

It is easy to see that the indices $\bar{r}(j)$, where $j \in C^-(v_i)$ for (27) and $j \in C^+(v_i)$ for (28), determine the most violated constraints, if any exists. We call the inequalities (27)–(28) SI1-FLIP, standing for Separated I1-FLIP.

5. Computational experiments

In this section we describe the computational experiments in which we tried to assess the effectiveness of the local search inequalities. We ran some tests for the TSP problem, Max-Cut and Max-SAT. The results are discussed in Sections 5.1–5.3, respectively. In our tests we used CPLEX version 12.1 as the LP-solver, called by a general-purpose branch-and-cut code that we developed. In general, the branch-and-bound search strategy was depth-first, while the branching variable was the most fractional. An exception to this branching rule, that turned out to be quite effective, was adopted for the Max-Cut problem, and is described in Section 5.2.

In order to study the effectiveness of LSIs, we have used two fundamental measures: the running time and the total number of nodes explored in the search-tree. Both measures were evaluated with and without the addition of LSIs. Clearly, the two measures are correlated, since to a decrease in the number of nodes explored should correspond a decrease in the running time, which is, ultimately, our main goal. However, there is a trade-off to evaluate: in some cases the addition of LSIs can lead to fewer nodes explored, but at the same time it increases the computational work at each node. As a result the overall running time can increase. When this phenomenon happened, we noticed that most of the times the running time did not increase by much, while the number of nodes was quite smaller. Hence, there is the hope that by fine-tuning all the work done at each node other than the LP-solution (e.g., by avoiding similar/identical re-computations at different nodes, for example with the use of suitable global data structures or incremental computations from a node to its descendants, etc.) the running time can be further decreased. We did not perform this type of fine tuning, but have elected to report both the running time and the number of nodes explored in our computational experiments.

The goal of the experiments is, loosely speaking, to "see if LSIs work". Hence the way we present the results reflects this goal. If (T_0, N_0) are the running time and number of nodes explored without LSIs, and (T, N) are the same parameters after the addition of LSIs, it makes sense adding LSIs if either $T < T_0$ or $N < N_0$, and this independently of the actual values of T_0, N_0, T, N. We have therefore decided to normalize all values so that T_0, N_0, T, N, being irrelevant, are not reported, while we report the percentages p_T and p_N of increment or decrement of T and N with respect to T_0 and N_0, namely

$$p_T = \frac{T - T_0}{T_0} \times 100, \qquad p_N = \frac{N - N_0}{N_0} \times 100.$$

This normalization has a practical advantage. In particular, it allows us to run the experiment across different hardware platforms (clearly, after making sure that we use the same CPLEX version, and the same random seeds, so that all runs would be identical on different machines except for the actual running time). Due to the number of experiments and the overall computing time needed, this parallelism allowed us to complete our experiments in a shorter amount of time.

In general, for each problem, we consider a set of instances that are first solved without LSIs (thus, returning the base reference values of T_0's and N_0's) and then solved again with the addition to the model of different type of LSIs (individually or together). A first table reports the percentage of increment/decrement of each LSI with respect to the base reference values. In a second table, we report, for each LSI, how many times the model with that particular LSI turned out to yield the minimum running time and/or number of search nodes. We call this the *number of wins* for that strategy, so that this table makes it easier to appreciate which is the best overall strategy, among not having LSIs, or having a particular type of LSIs.

Table 2
TSPLIB instances solved with different LSIs.

		pr124	ch130	d198	kroA100	kroE100	kroA150	kroB150	bier127	pr136	pr152
SWAP	p_T	−57.1	−50.0	+8.1	0	−42.9	−12.5	+39.6	−46.0	+20.1	−18.7
	p_N	−75.0	−66.6	−17.7	−39.6	−49.6	−40.4	−9.5	−67.2	−22.4	−43.6
2OPT (4)	p_T	−64.3	−27.8	−11.7	−66.7	−57.1	−58.3	−59.4	−50.7	−22.9	**−76.3**
	p_N	−72.2	−33.2	−17.5	−70.7	−61.0	−61.5	−63.3	−55.7	−25.9	−80.5
2OPT (5)	p_T	−21.4	−5.56	−8.1	−46.2	−42.9	−12.5	−43.0	−37.3	−5.5	−23.0
	p_N	−28.1	−9.16	−12.2	−48.7	−36.4	−16.3	−45.6	−42.4	−6.8	−26.2
2OPT (7)	p_T	**−71.4**	−44.4	+2.7	**−79.5**	**−71.4**	**−60.4**	−45.9	−71.0	−21.7	−69.1
	p_N	−78.6	−53.0	−12.2	−83.3	−68.5	−66.0	**−72.8**	−76.5	−24.0	−74.2
2OPT (8)	p_T	−50.0	−27.8	−16.2	−71.8	−57.1	−50.0	−53.1	−61.6	−30.1	−38.1
	p_N	−59.2	−33.2	−20.4	−75.3	−62.2	−53.1	−56.9	−66.6	−23.3	−43.4
2OPT (9)	p_T	−64.3	−44.4	−15.3	−71.8	−57.1	−45.8	−55.1	**−79.1**	−28.7	−51.8
	p_N	−68.0	−49.2	−20.4	−76.3	−60.5	−52.4	−59.2	**−82.6**	−20.5	−57.4
2OPT (10)	p_T	−57.1	**−61.1**	**−38.7**	−59.0	**−71.4**	−39.6	**−62.8**	−64.8	**−64.5**	−56.8
	p_N	−71.1	−73.3	**−46.1**	−76.6	−71.9	−53.1	−71.7	−75.5	**−55.5**	−67.7
All	p_T	**−71.4**	−50.0	−0.9	−66.7	−42.9	−33.3	−19.3	−48.1	−43.8	−66.9
	p_N	**−87.6**	**−78.3**	−38.4	**−88.3**	**−72.3**	**−68.2**	−66.7	−79.2	−53.0	**−85.6**

Table 3
With LSI vs without LSI.

LSI	Time	Nodes
SWAP	7/10	10/10
2OPT (4)	10/10	10/10
2OPT (5)	10/10	10/10
2OPT (7)	9/10	10/10
2OPT (8)	10/10	10/10
2OPT (9)	10/10	10/10
2OPT (10)	10/10	10/10
All	10/10	10/10

Table 4
Wins for different LSIs.

LSI	Time	Nodes
None	0/10	0/10
SWAP	0/10	0/10
2OPT (4)	1/10	0/10
2OPT (5)	0/10	0/10
2OPT (7)	4/10	1/10
2OPT (8)	0/10	0/10
2OPT (9)	1/10	1/10
2OPT (10)	**5/10**	2/10
All	1/10	**6/10**

5.1. TSP

To test our LSIs for the TSP, we used a Branch-and-Cut code which was run on ten instances from the TSPLIB [19]. The instances were randomly chosen among those of small/moderate size. The basic reference ILP model was (2)–(3) i.e., degree constraints and subtour inequalities. In Table 2 we report the computational results for our seven families of LSIs over the ten TSPLIB instances. Each row is labeled with one type of LSI, and is relative to the basic model with the addition of the corresponding LSI. Each family of LSIs is added to the model by a separation procedure which determines the violated inequalities. The first row refers to SWAP LSI, while the other rows are labeled each with the reference to the relative 2OPT LSI (from 2OPT (4) to 2OPT (10)). The last row, labeled "all", corresponds to the basic model with the addition of all the above LSIs together.

For each column, we highlight in boldface the entries yielding the best running time and number of nodes in the search tree. It can be seen that there is always a LSI that beats the basic TSP model without LSIs. Furthermore, a look at the table by rows shows that the addition of each LSI would, on average, yield an improvement over the basic TSP model.

This result is detailed in Table 3, where we report for each family of LSIs how many times the basic model plus the corresponding LSIs outperforms the model without the LSIs. It can be seen that each type of LSIs yields an improvement over most instances. In fact, most LSIs yield an improvement over *all* instances.

In Table 4 we report, for each type of LSI, the number of times it yields the best performance over the 10 instances (notice that, since different types of LSIs can yield the same performance, the sum of these values over all the LSIs can exceed 10).

Table 5
Max Cut random instances.

		G(30, 0.5)		G(30, 0.75)		G(40, 0.5)	
		p_T	p_N	p_T	p_N	p_T	p_N
No LSI/B_C	min	−31.5	−51.6	−8.6	−0.8	−13.2	−11.2
	max	+10.0	0.0	+4.3	+0.4	+30.9	0.0
	avg	**−4.1**	**−3.9**	−1.2	−0.1	+0.3	−1.9
1-FLIP/B_F	min	−12.5	−38.7	−22.6	−41.4	−25.8	−35.4
	max	+116.2	+41.9	+48.8	+10.8	+52.6	+42.1
	avg	+27.5	+1.5	−1.0	−26.8	+11.6	−3.9
1-FLIP/B_C	min	−23.3	−38.7	−32.7	−49.8	−32.1	−43.0
	max	+87.5	+27.9	+9.1	−19.8	+36.7	+12.9
	avg	+19.7	−3.8	+17.1	−39.9	**−0.3**	**−15.9**
I2-FLIP/B_F	min	−19.2	−38.7	−19.0	−47.2	−13.4	−30.2
	max	+150.0	+74.4	+62.5	−6.0	+65.9	+52.6
	avg	+48.4	+15.4	+11.6	−30.3	+23.0	+3.9
I2-FLIP/B_C	min	−12.3	−38.7	−60.2	−68.1	−27.8	−43.5
	max	+130.0	+37.2	−5.8	−38.7	+34.6	+15.5
	avg	+35.3	−0.9	**−31.8**	**−58.4**	+1.7	−15.8

Table 6
Max-Cut: the number of wins for different LSIs strategies.

	G(30, 0.5)		G(30, 0.75)		G(40, 0.5)	
	Time	Nodes	Time	Nodes	Time	Nodes
No LSI/B_F	7/20	6/20	0/20	0/20	3/20	1/20
No LSI/B_C	**13/20**	10/20	0/20	0/20	6/20	2/20
1-FLIP/B_F	1/20	5/20	1/20	0/20	1/20	2/20
1-FLIP/B_C	2/20	**13/20**	1/20	0/20	**7/20**	7/20
I2-FLIP/B_F	1/20	5/20	0/20	0/20	0/20	0/20
I2-FLIP/B_C	0/20	7/20	**18/20**	**20/20**	3/20	**11/20**

It can be noted that the best improvement in the running time is achieved by using LSIs 2OPT (10) alone (five times out of ten), while the second best are 2OPT (7). With respect to the number of explored nodes, the best savings are achieved when all the LSIs are included together to the model. In this case, as one can deduce from last row of Table 2, the average number of nodes with the LSIs is less than a third of the number without the LSIs.

5.2. Max-Cut

In a second run of tests we considered the Max-Cut problem, in which the basic model was the maximization of $\sum_{ij\in E} x_{ij}$ under the triangle inequalities (18)–(21).

For this problem we introduced also an *ad-hoc* branching strategy, alternative to the standard choice of branching on the most fractional variable. This strategy has combinatorial motivations, described below. Let us call B_F the standard branching on fractional variables and B_C the new branching strategy. In our computational tests, the basic model for Max-Cut was extended in five ways, i.e., one by using the B_C strategy on the model without LSIs, and the other four corresponding to all the combinations of adding to it one of the two LSIs and using B_F or B_C.

A branching scheme for Max-Cut. Consider a node of the search tree, and let F_0 and F_1 be the set of variables that can only assume the values 0 and, respectively, 1 in any feasible solution at the node. Let $F = F_0 \cup F_1$. We recall that in the Max-Cut model the variables correspond to binary relations between the nodes, i.e., $x_{ij} = 0$ if i and j are on the same shore of the cut, and $x_{ij} = 1$ otherwise. Some of the variables in F have been fixed to their values by the branching constraints leading to the node, while others are forced to their values by logical implications. In particular, consider the connected components of the graph $G_F := (V, F)$ (note that G_F is not necessarily a subgraph of G, but rather of the complete graph over V). It is easy to see that each connected component of G_F is a clique. Note that we can explicitly add the constraints $x_{ab} = 0/1$ for all the variables in F that are not explicitly fixed by branching constraints. This is better from a practical point of view, even if these constraints are in fact already implied by constraints (18)–(21).

For the branching rule, it makes sense to choose the fractional variable whose setting will force the largest number of unfixed variables to assume fixed values. The first time we branch, only one variable becomes fixed and so we choose the most fractional variable, as usual. At later branching decisions, however, we use a strategy similar to Prim's strategy for the Minimum Spanning Tree problem. Namely, at the generic branching, we have only one connected component in F, say C, of size ≥ 1, while all other components are isolated nodes. By branching on a variable x_{ab} such that $ab \in \delta(C)$, there will be $|C|$ variables that will assume fixed values, since C will be added a new node and become a clique of size $|C| + 1$. The branching rule is therefore the following: *among the fractional LP variables x_{ab} with $ab \in \delta(C)$, choose the most fractional.*

In our tests, we generated some random instances in which we set the density (i.e., probability that an edge belongs to the instance) to $\frac{1}{2}$ and $\frac{3}{4}$ (this implies an average degree of $\frac{1}{2}(n-1)$ and $\frac{3}{4}(n-1)$ respectively). In Table 5 we report the results

Table 7
2SAT results.

Instance	1-FLIP		I1-FLIP		SI1-FLIP	
	p_T	p_N	p_T	p_N	p_T	p_N
sz60.300.1.cnf	−6.8	−35.1	−60.2	**−90.4**	**−81.4**	**−90.4**
sz60.300.2.cnf	0	−30.4	−50.0	−87.0	**−83.3**	**−87.1**
sz60.300.3.cnf	+40.0	−6.7	+20.0	**−68.0**	**−40.0**	−67.9
sz60.300.4.cnf	+28.6	−12.7	−28.6	**−84.1**	**−71.4**	−78.9
sz60.300.5.cnf	0	−41.4	−47.4	−86.7	**−78.9**	**−86.8**
sz60.300.6.cnf	0	−34.5	−47.6	**−84.0**	**−76.2**	−83.9
sz60.300.7.cnf	−25.0	−42.7	−53.1	**−88.1**	**−78.1**	**−88.1**
sz60.300.8.cnf	0	−32.2	−59.2	**−89.1**	**−80.3**	−89.0
sz60.300.9.cnf	−37.1	−54.4	−62.9	−91.1	**−83.1**	**−91.2**
sz60.300.10.cnf	−6.3	−34.3	−31.2	−83.8	**−75.0**	**−84.1**
sz60.400.1.cnf	+48.3	−16.4	−18.1	**−89.8**	**−68.7**	**−89.8**
sz60.400.2.cnf	−4.5	−51.4	−78.4	**−96.9**	**−91.3**	**−96.9**
sz60.400.3.cnf	+41.6	−23.2	−57.3	**−93.4**	**−82.6**	**−93.4**
sz60.400.4.cnf	+7.0	−34.0	−46.9	**−90.6**	**−77.2**	**−90.6**
sz60.400.5.cnf	−27.9	−54.0	−74.1	**−95.8**	**−88.8**	**−95.8**
sz60.400.6.cnf	−23.7	−51.3	−76.3	**−95.8**	**−88.8**	**−95.8**
sz60.400.7.cnf	−26.7	−64.9	−78.3	**−95.7**	**−90.1**	**−95.7**
sz60.400.8.cnf	0	−45.8	−72.0	**−95.2**	**−87.3**	−95.1
sz60.400.9.cnf	−13.5	−37.5	−76.4	**−95.5**	**−88.9**	**−95.5**
sz60.400.10.cnf	−43.8	−62.5	−80.0	**−96.9**	**−91.0**	−96.8
sz60.500.1.cnf	+90.3	−26.5	−43.3	**−95.3**	**−77.7**	**−95.3**
sz60.500.2.cnf	+43.1	−39.0	−83.0	**−98.0**	**−92.7**	**−98.0**
sz60.500.3.cnf	+24.6	−36.7	**−98.9**	**−98.1**	−93.5	**−98.1**
sz60.500.4.cnf	+4.9	−47.3	−83.0	−98.3	**−93.4**	**−98.4**
sz60.500.5.cnf	−11.7	−51.3	−91.5	**−98.9**	**−95.8**	**−98.9**
sz60.500.6.cnf	−6.1	−56.6	−85.8	**−98.0**	**−93.1**	**−98.0**
sz60.500.7.cnf	−14.6	−49.6	−81.2	**−97.7**	**−91.8**	−97.6
sz60.500.8.cnf	−68.0	−84.4	−97.7	**−99.6**	**−98.8**	**−99.6**
sz60.500.9.cnf	−26.3	−59.4	−94.5	**−99.3**	**−97.2**	**−99.3**
sz60.500.10.cnf	−54.9	−80.6	−78.3	**−97.8**	**−92.0**	−97.7

over 60 instances. There are 40 instances with 30 nodes, half with density 0.5, and half with density 0.75. Furthermore, there are 20 instances with 40 nodes and density 0.5, which was the largest size which the model could tackle in a reasonable amount of time on our machine (i.e., within about one hour per instance). The table has three columns, one per each instance family, and five rows, each depending on the LSIs and the branching strategy used. The reference model is the basic ILP model with standard branching B_F. For each instance family and combination of LSI/branching, we report the minimum, average, and maximum of the relative increment of time and nodes of the search tree with respect to the reference model. In each column we highlighted the best relative increment. Since there are only $O(n)$ 1-FLIP LSIs, there is no need for a separation algorithm and they were all added to the model. The I2-FLIP LSIs, on the other hand, were separated.

In Table 6 we consider each combination of LSIs and branching rule, and see how many times, over all instances, it yields the best result. It appears that for this particular problem, the best results are achieved by using the branching strategy B_C, and that LSIs become more effective as the size of the graphs grows (either because of a larger number of edges or nodes or both). For the smallest graphs, the best results as far as running time is concerned are obtained by using the basic model without LSIs, while the greatest savings in nodes are achieved by using the 1-FLIP LSIs. For the 30-nodes graph at density 0.75, the best running times and search nodes are obtained by using I2-FLIP LSIs. For the 40-nodes graphs, the best running times are obtained with 1-FLIP LSIs, while the fewer search tree nodes are obtained with the I2-FLIP LSIs.

5.3. Max-SAT

The experiments for Max-SAT problems were run on instances taken from the literature. In particular, they are instances from "The Second Evaluation of Solvers" (MaxSAT-2007), an affiliated event of the Tenth International Conference on Theory and Applications of Satisfiability Testing [20]. There are 2SAT and 3SAT instances, with either $n = 40$ or 60 variables over $m = 300$, 400, 500 and 600 clauses (each file name specifies the values of n and m for an instance. In our table, we have reported the original file names, as from [20]).

In Table 7 we report the results of LSIs for the 30 instances of 2SAT. In the comparisons, the ILP reference model for SAT is (22). As it can be observed, the inequalities SI1-FLIP allow the greatest savings in running times on all but one instance (for which I1-FLIP yields the best running time). For some instances, such as `sz60.500.8`, the running time and number of nodes are almost two orders of magnitude smaller than when LSIs are not added to the model.

In Table 8 we report a summary of the LSIs performance over all 30 instances, grouped by instance size. It can be seen that I1-FLIP and SI1-FLIP allow similar savings with respect to the number of nodes, but using SI1-FLIP is better as far as the running time is concerned.

Table 8
2SAT aggregate results.

		1-FLIP		I1-FLIP		SI1-FLIP	
		p_T	p_N	p_T	p_N	p_T	p_N
sz60_300	min	−37.1	−54.4	−60.2	−90.4	−81.4	−90.4
	max	+40.0	−6.7	+20.0	−68.0	−40.0	−67.9
	avg	−0.7	−32.4	−42.0	**−85.2**	**−74.8**	−84.7
sz60_400	min	−43.8	−64.9	−78.4	−89.8	−91.3	−89.8
	max	+48.3	−16.4	−18.1	−89.8	−68.7	−89.8
	avg	−4.3	−44.1	−65.8	**−94.6**	**−85.5**	**−94.6**
sz60_500	min	−68.0	−84.4	−97.7	−99.3	−92.7	−99.3
	max	+90.3	−26.5	−43.3	−95.3	−77.7	−95.3
	avg	−1.8	−53.2	−82.4	**−98.1**	**−92.6**	**−98.1**

Table 9
3SAT results.

	1-FLIP		SI1-FLIP	
Instance	p_T	p_N	p_T	p_N
cnf3.40.400.907721	+18.75	+0.64	+9.38	**−5.99**
cnf3.40.400.907722	+14.29	−7.74	+4.76	**−14.87**
cnf3.40.400.907723	+46.43	+13.08	+3.57	**−14.77**
cnf3.40.400.907724	**−29.89**	**−41.22**	+13.22	−14.56
cnf3.40.400.907725	+14.46	−4.29	+9.04	**−9.84**
cnf3.40.400.907726	+24.18	+9.52	+10.99	**−10.15**
cnf3.40.400.907727	+9.78	**−10.37**	+8.70	−8.66
cnf3.40.400.907728	**−11.25**	**−25.80**	+12.50	−7.65
cnf3.40.400.907729	+31.58	+8.94	+10.53	**−1.66**
cnf3.40.400.907730	+16.67	**−4.49**	+16.67	−3.33
cnf3.40.600.359121	+10.64	−11.10	+9.80	**−34.17**
cnf3.40.600.359122	+0.93	−19.64	+14.81	**−20.44**
cnf3.40.600.359123	+27.33	+3.16	+4.53	**−33.60**
cnf3.40.600.359124	+49.23	+18.82	+12.31	**−11.17**
cnf3.40.600.359125	+22.04	−1.83	+7.95	**−34.61**
cnf3.40.600.359126	+47.21	+21.39	+0.97	**−40.63**
cnf3.40.600.359127	+17.88	−6.09	+12.73	**−20.42**
cnf3.40.600.359128	+72.45	+38.42	+17.14	**−12.03**
cnf3.40.600.359129	**−14.69**	−32.89	+4.88	**−35.07**
cnf3.40.600.359130	+35.62	+4.38	+15.07	**−18.82**

Table 10
3SAT aggregate results.

		LC1		LC3	
		p_T	p_N	p_T	p_N
cnf3.40_400	min	−29.89	−41.22	+3.57	−14.87
	max	+46.43	+13.08	+16.67	−1.66
	avg	+13.50	−6.17	+9.93	**−9.15**
cnf3.40_600	min	−14.69	−32.89	+0.97	−40.63
	max	+72.45	+38.42	+17.14	−11.17
	avg	+26.86	+1.46	+10.02	**−26.10**

In Table 9 we report the results for the 20 instances of 3SAT. Due to the very large size of the LP when inequalities I1-FLIP are added to the model, we have only considered the LSIs 1-FLIP and SI1-FLIP. For 3SAT the results are quite worse than for 2SAT. In particular, while the addition of LSIs causes almost invariably a decrease in the number of search tree nodes, only for 3 out of 20 instances there is an improvement in the running time when LSIs are added to the model. While this always happens with 1-FLIP LSIs, it can be seen from Table 10 that, on average, SI1-FLIP are better than I1-FLIP with respect to the running times. However, the best model for 3SAT is the one without LSIs.

6. Conclusions

In this paper we have described a new type of inequalities, the local search inequalities, which impose local optimality conditions that all global optima must satisfy. LSIs are not valid inequalities in the usual sense, but they can be added to ILP models to improve their performance. We have seen that, for some ILP models, the number of search nodes explored, as well as the total time needed to obtain the optimal solution, can be reduced with the use of LSIs.

LSIs are very general and "formulation independent", i.e., the same LSI can be added to many different formulations as long as such formulations include the variables of the LSI.

Our computational experiments have shown that the addition of LSIs can be quite beneficial in improving some basic, simple, ILP models for the optimization problems that we considered. To better assess the impact of LSIs in branch-and-bound schemes it would be important to embed LSIs in state-of-the-art branch-and-cut codes for TSP, Max-Cut and Max-SAT, where the underlying ILP models are more sophisticated than the ones we considered.

In this paper we have limited ourselves to laying out the basic ideas and performing some preliminary tests. We believe that presenting a theoretical idea and 'engineering' the same idea into an existing code are two separate tasks. Of course the idea must have some good computational behavior, or otherwise it is useless, and so we have carried out the tests with this purpose, but there may be a long way before improving and tuning the idea. This should be matter of future research.

References

[1] G.L. Nemhauser, L.A. Wolsey, Integer Programming and Combinatorial Optimization, John Wiley & Sons, New York, NY, USA, 1988.
[2] M. Padberg, G. Rinaldi, A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems, SIAM Rev. 33 (1) (1991) 60–100.
[3] E. Aarts, J.K. Lenstra (Eds.), Local Search in Combinatorial Optimization, John Wiley & Sons, London, 1997.
[4] W. Michiels, E. Aarts, J. Korst, Theoretical Aspects of Local Search, Springer, Berlin, 2007.
[5] N.J. Hooker, Logic-based methods for optimization, in: A. Borning (Ed.), Principles and Practice of Constraint Programming, in: Lecture Notes in Computer Science, vol. 874, 1994, pp. 336–349.
[6] I. Grossmann, J.N. Hooker, R. Raman, H. Yan, Logic cuts for processing networks with fixed charges, Comput. Oper. Res. 21 (1994) 265–279.
[7] M. Fischetti, J.J. Salazar Gonzalez, P. Toth, Solving the orienteering problem through branch-and-cut, INFORMS J. Comput. 10 (1998) 133–148.
[8] M. Jünger, G. Reinelt, G. Rinaldi, M.O. Ball, T.L. Magnanti, C.L. Monma, G.L. Nemhauser, The traveling salesman problem, in: Network Models, Handbooks in Operations Research and Management Science, vol. 7, North-Holland, Amsterdam, The Netherlands, 1995, pp. 225–330.
[9] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys (Eds.), The Traveling Salesman Problem, John Wiley & Sons, Chichester, 1985.
[10] T. Öncan, I.K. Altinel, G. Laporte, A comparative analysis of several asymmetric travelling salesman problem formulations, Comput. Oper. Res. 36 (2009) 637–654.
[11] D.S. Johnson, L.A. McGeoch, The traveling salesman problem: A case study in local optimization, in: E. Aarts, J.K. Lenstra (Eds.), Local Search in Combinatorial Optimization, John Wiley & Sons, London, 1997, pp. 215–310.
[12] M. Grötschel, M.W. Padberg, On the symmetric travelling salesman problem 1: inequalities", Math. Program. 16 (1979) 265–280.
[13] M.R. Garey, D.S. Johnson, L. Stockmeyer, Some simplified NP-complete graph problems, Theoret. Comput. Sci. 1 (3) (1976) 237–267.
[14] F. Barahona, A.R. Mahjoub, On the cut polytope, Math. Program. 36 (1986) 157–173.
[15] T. Bonato, M. Jünger, G. Reinelt, G. Rinaldi, Lifting and separation procedures for the cut polytope, Math. Program. 146 (2014) 351–378.
[16] C. De Simone, G. Rinaldi, A cutting plane algorithm for the max-cut problem, Optim. Methods Softw. 3 (1994) 195–214.
[17] G. Lancia, P. Serafini, An effective compact formulation of the max cut problem on sparse graphs, Electron. Notes Discrete Math. 37 (2011) 111–116.
[18] F. Rendl, G. Rinaldi, A. Wiegele, Solving Max-Cut to optimality by intersecting semidefinite and polyhedral relaxations, Math. Program. 121 (2010) 307–335.
[19] G. Reinelt, TSPLIB—a traveling salesman problem library, ORSA J. Comput. 3 (1991) 376–384.
[20] http://www.maxsat.udl.cat/07/index.html.