Modeling and results of Satellite Ground Support Optimization

Sanjay Kumar
University of Texas in Dallas, Dallas Texas USA
skumar@utdallas.edu
and
Tapan P Bagchi
Indian Institute of Technology Bombay, Mumbai 400076 India
bagchi@iitb.ac.in

Abstract

This paper will present the methods adapted to model low earth orbiting (LEO) spacecraft support and an actual implementation of this approach. The results of a comparison of actual human schedulers' performance and the support schedules developed by using the proposed modeling approach will be presented. Indications based on two years of performance tracking of the new approach are that modeling optimization yields up to half an hour/day of extra support for remote sensing satellites, a very significant gain when one considers the high cost of putting a LEO satellite in orbit.

Key words: LEO spacecraft, Ground station support, Product mix Allocation, Optimization, Nonlinear objectives, Nonlinear constraints, Genetic algorithms, Meta-heuristics

1. What is a Visibility Clash?

Orbiting once every 100 or so minutes at a 800 km height, low orbit (LEO) spacecraft now form a critical infrastructure for remote sensing, natural resource management, crop estimation, flood control, communication, and space research support [11]. This paper addresses the optimal allotment of ground station support time to passing LEO spacecraft when their visibilities clash. The problem is NP-complete and more complex than classical product mix determination since the products here may have different and arbitrarily defined profitability profile. Further, opportunity windows dynamically generate clashing topologies. Decision makers in *satellite mission support management* [1, 2], *motion picture exhibition* [6, 20] and *advertising* (rather distinctively in *e-commerce*) [21] now routinely face its variants.

V*isibility clashes* are incidents when two or more spacecraft passing over a ground station have overlapping visibilities and ground station resources must be apportioned equitably among the spacecraft so as to generate maximum value in the mission. This paper describes meta-heuristic methods to optimally resolve visibility clashes. The methods apply both to TTC (telemetry, tracking and command) and PL (payload) operations support. Initially we use a simple, linear objective function to set the stage for the procedure. Subsequently we describe a method to *derive* the applicable non-linear optimization function—to help resolve visibility clash with

realistic constraints and objectives. Such a payoff function can capture preferences, penalties, soft constraints, mission priorities and other factors as articulated by space mission scientists. This leads to a general solution to the clash resolution.

Broadly defined, a "visibility clash" is said to occur when LEO spacecraft have either overlapping visibilities at a ground station or their respective LOS (Loss of Signal) and AOS (Acquisition of Signal) difference is less than the station reconfiguration time. Nonclashing visibilities may be assigned tasks by priority-dispatch rules [17]. However, "clash resolution" must find the optimum *apportioned* support schedule that maximizes the total value or utility generated from supports provided to each clashing spacecraft. In this process some spacecraft may receive no support at all, yet the total schedule is optimal. As for the character of the objective functions optimized, one with a constant payoff rate is pertinent in PL support optimization. A function with a diminishing marginal payoff rate is apt in resolving TTC clashes.

Clearly, the task grows in complexity as the number of spacecraft in orbit grows. Table 1 displays a fleet of six Indian LEO spacecraft that must be supported together, with possibility of several more whose missions must join this fleet.

Table: 1 Sample Spacecraft Fleet in Orbit

SPACECRAFT	*ORBIT #*	*LAUNCH DATE*	*LOCAL TIME*	*PAY LOADS*
IRS – 1C	808 / 825	28-12-1995	9:48	PAN, WIFS LISS – 3
IRS – 1D	736/821	29-09-1997	10:23	PAN, WIFS LISS – 3
IRS – P3	807 / 825	21-03-1996	8:53	XRAY, MOS CBT
IRS- P4	711/728	26-05-1999	11:57	OCM, MSMR
IRS – P6	808 / 825	17 – 10 –2004	10:34	LISS – 4, AWIFS,
TES	523 / 585	22 – 10 –2001	10:52	LISS – 3,SSR PAN

2. Optimization of Visibility Clash: The Challenges

The optimal allotment of ground station support time to passing low orbit spacecraft is an NP-complete problem [23]. It is more complex than product mix determination in manufacturing management since the "products" here may have different and arbitrarily defined profitability profile. Further, clashing topologies generate the opportunity windows dynamically. Besides spacecraft mission support management [1, 2], decision makers in *motion picture exhibition* [6, 20] and *advertising* (rather distinctively in *e-commerce*) [21] also face variants of this problem routinely.

The fundamental time allocation problem may be characterized as below.

- The opportunity window (machine time) is open for a specified period within which the resources must be allocated (to products to be made) so as to maximize the total profit contributed by all resources allocated.
- Products are perishable. They have no salvage value. Each product must be marketed (consumed) as produced. There is no inventory.
- Each product has its own marketing opportunity.
- In general, the market opportunities of two or more products may clash,

forcing the enterprise to produce and market only one product at one instant of time.
- If a product is produced, a specified minimum quantity of it must be produced.
- Production of a product once started cannot be interrupted (no job splitting)
- For each product there is a maximum quantity that can be produced regardless of the market opportunity that may exist for it.
- A finite switchover time is required to reconfigure the machine from producing one product to another. This time is machine dependent and independent of the production quantities involved.
- The machine itself is available for a specified period.
- An arbitrary profit function determines how much profit is contributed by each product.

The above problem may initially be patterned after parallel machine scheduling [5, 19]. A set of N jobs is to be scheduled on M parallel machines. Each job j, ($j = 1,\ldots, N$), must be processed without interruption during a period of length OPD_j, that is, the minimum *obligation* period for which the job will be occupying the machine, if scheduled. A machine can handle no more than one job at a time, and is continuously available. Each job j has a *release date* r_j and a *due date* d_j. The goal is to find an optimal feasible schedule—a set of start and end times—such that the capacities, availability and time limit constraints are met, and a given objective function is optimized. As said, this problem is NP-complete [14].

Spacecraft task scheduling and sequencing has received increasing attention in the past decade as the application of spacecraft in remote sensing, crop estimation, rescue missions, mapping, defense, and numerous other applications has flourished. An activity-based scheduler exploiting AI methods was used to represent constraints and for searching good schedules to determine near-optimal long term scheduling of Hubble telescope observations [19]. The method used heuristic logic and hill-climbing schedule repair schemes to determine a hierarchical ordering of activities, in order to schedule the most constrained activities first. A system for NASA's Terriers satellite based on dispatching rules was implemented in 1992 without the explicit consideration of optimality [9]. GREAS, a scheduling and mission planning tool using task mapping-to-resource logic utilizes a constraint satisfaction approach [8, 13]. This system first creates a representation involving tasks, resources, events and constraints. It then searches for a solution (a feasible schedule) using heuristic search with constraint propagation.

Agnese and Brousse [2] present a search method to apply depth first and branch and bound methods, and greedy search to determine task allocations to spacecraft visibility windows. The method is shown to yield solutions within reasonable time, but it is approximate and thus suboptimal. A commercial system called ILOG aims as satisfying critical constraints only [24]. Tasks are allocated in ILOG using integer programming on a yes/no rather than fractional coverage (to allow true optimality)

basis. Other similar suboptimal spacecraft support schedules have been produced by [18]. Among the recent noteworthy works in this area, Wolfe and Sorensen [23] compare priority dispatch, a look ahead heuristic and genetic algorithms to assign tasks to earth orbiting satellites.

In this paper a special case of this problem is considered first. The general visibility clash resolution problem is addressed next.

3. The Parallel Machines Model: An Exact Solution

A special case of the support scheduling problem may be formulated and exactly solved as follows. All of the support stations are assumed to be identical and modeled as parallel machines. At any point in time, the decision maker may decide whether or not to schedule a particular job (a support task). However, once scheduled, the processing period is restricted to be *at least* OPD_j. Also, scheduling decisions are made keeping in mind the constraints on the release and due dates (the appearance and the disappearance of the spacecraft). Further, the jobs may contribute differently to the total profitability when scheduled at different points in time. The objective is to schedule all jobs available such that the cumulative profit is maximized over the planning horizon. Lastly, the decision maker may decide to schedule only a subset of all the jobs available.

A Time-Indexed Formulation of the Problem

A *time-indexed formulation* [18, 22] may be used to solve such problems analytically. This uses the idea of dividing the planning horizon [0, …, W] into W discrete intervals of unit length such as weeks. A binary decision variable is used, which equals 1 if job j is scheduled for i weeks *beyond its obligation period* starting in week w of the planning horizon, and 0 otherwise. This action incorporates the obligation period constraint in the definition of the decision variable itself. This formulation may be generalized to encompass such extensions as different machine capacities, precedence constraints, and job-specific due dates.

The time-indexed formulation highlights some key differences between the current problem and typical machine scheduling problems. First, all jobs do not have to be processed in the current problem, while all jobs must be scheduled in machine scheduling problems. Accordingly, this formulation helps the decision maker resolve two critical issues: *choice* of "which" jobs to schedule, and deciding *processing durations* of the chosen jobs. Second, typical decision making goals prevalent in machine scheduling problems are turnaround, timeliness, and throughput. In contrast, the current formulation offers a situation in which the scheduling decisions directly affect profitability, perhaps a more appropriate decision objective.

Solving the Problem

A profit function, or a similar utility measure, could be defined corresponding to each value of the decision variable. For example, corresponding to the value of a

particular binary decision variable being 1, the function would denote the profit the decision maker would make if job *j* is scheduled for *i* weeks *beyond its obligation period* starting in week *w* of the planning horizon. This operationalization of the profits would simplify the solution procedure considerably since it would be possible to compute them independently of the algorithm for generating the optimal schedule. Based on such factors, the time-indexed formulation would be as follows.

Maximize Cumulative Profits

Decision variables are Support lengths, processing times or *duration* allocated to the individual products

Subject to

1. Continuity constraints
2. Machine allocation constraints
3. Release- and due-dates constraints, and
4. Binary constraints on the decision variables

Constraint 1 would ensure that a job is scheduled in only consecutive weeks (i.e., no job-splitting), when that is the requirement. Also, this would allow a job not to be scheduled at all, if it is not profitable to do so. The next constraint would restrict the total number of jobs scheduled at any instance of the planning horizon to *M*, the total number of machines available. In doing so, it would sum up all the jobs that are released earlier than or at the point of time under consideration. Constraint 3 would contain indexing constraints to ensure feasibility according to release and due dates. Constraint 4 would define the decision variable to be a binary variable.

The above problem may be formulated as an integer/linear program [6] and then, by exploiting the underlying special structure of the constraints (network property) it may be solved as a linear program. This approach greatly reduces solution time.

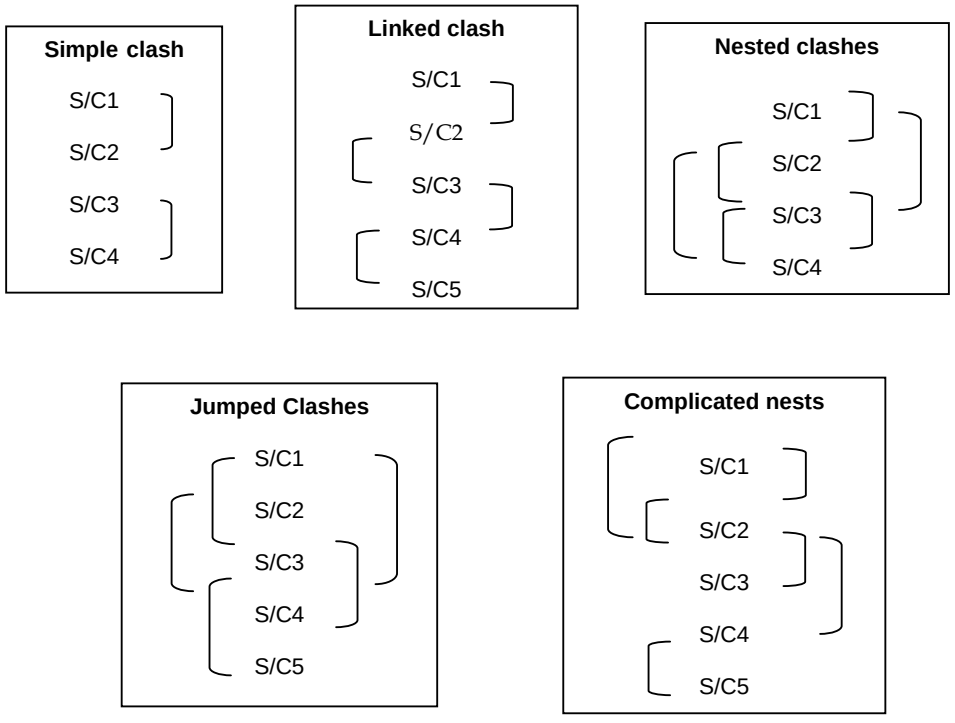
4 A Variation: Optimal Resolution of Spacecraft Visibility Clashes

Consider now the following variation of the parallel machine scheduling problem. As stated at the outset of this text, remote sensing spacecraft are typically low earth orbiting and these must remain in routinely executed support (communication) with ground stations located at different points on the globe. Table 2 displays a list of sample support stations in a network. As mentioned above, *visibility clash* (akin to a market opportunity clash) occurs when several spacecraft simultaneously passing over a ground station equipped with only a single "chain" (an antenna connected with appropriate electronic infrastructure) have either overlapping visibilities, or the relevant LOS (*Loss of Signal* of spacecraft #1) and AOS (*Acquisition of Signal* of spacecraft #2) difference is less than the station reconfiguration time. The challenge is to determine the optimum support schedule (i.e., how much support to give to which satellite) that maximizes the total profit (utility) generated from all supports executed—within a single clash situation.

Table 2: Sample Ground Support Stations in a Typical Network

Station	*Capability*	*Support Duration*
Bangalore (2 chains)	All operations; Reconfiguration time 3 minutes	Throughout mission life (All 24 hours)
Lucknow (2 chains)	All operations; Reconfiguration time 3 minutes	Throughout mission life (All 24 hours)
Mauritius (1 Chain)	All operations; Reconfiguration time 3 minutes	Throughout mission life (04:00 U T to 10:00 U T)
Bears Lake (2 chains)	All operations; Reconfiguration time 15 minutes	Throughout mission life (All 24 hours)
Biak (1 chain)	TM, TC	Throughout mission life (22:00 UT to 05:00 UT & 11:45 UT to 14:15 UT)

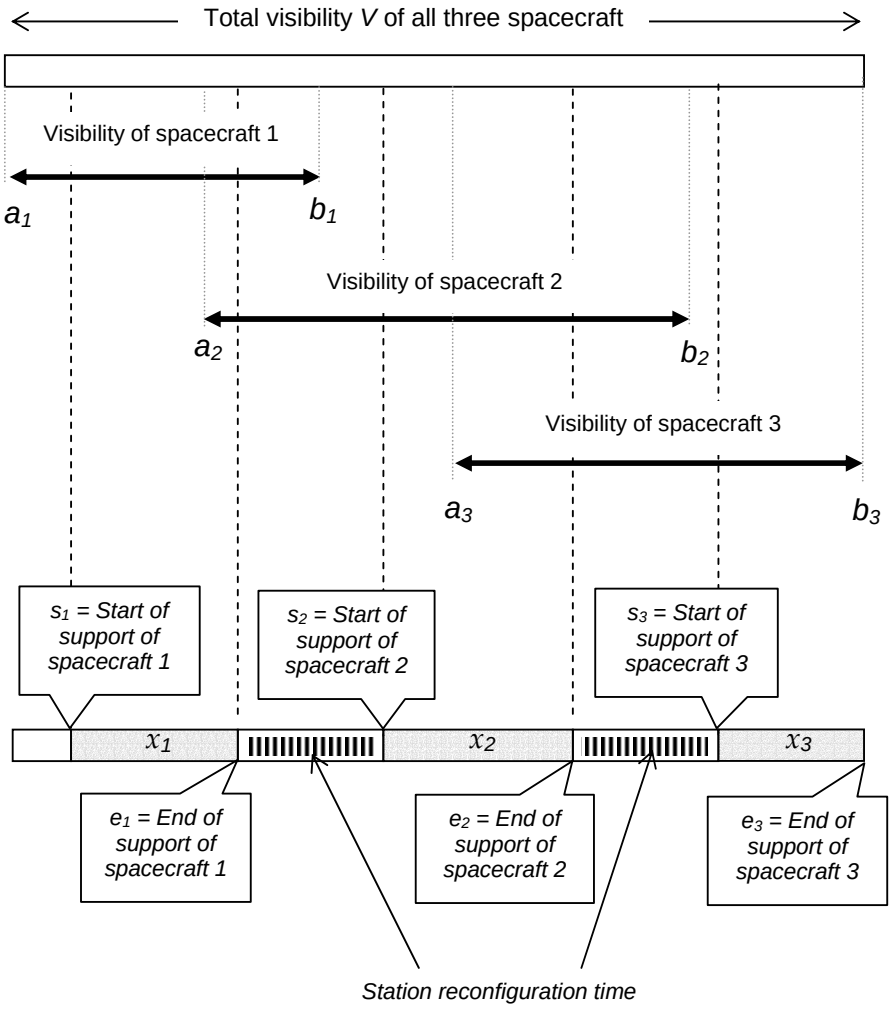
Figure 1 Variants of Visibility Clash Topologies



Complexities enter in this problem from two sources. First, visibility clashes become numerous when the spacecraft are a dozen or more in number because then the patterns in which these revolve may have large variety. This may greatly expand the solution search space. Indeed, note that the topologies shown in Figure 1 develop dynamically and must be handled precisely for optimal ground station support—possibly several dozen times every hour as the LEOs circle the globe. Second, the *objective function* to be optimized subject to various constraints (hard and soft) may be nonlinear and even discontinuous.

Figure 2 displays the general picture of a single occasion of clashing visibilities at a ground station. Notice the fat two-headed arrows that indicate the individual spacecraft's visibilities that overlap with each other. Note also the station reconfiguration periods interceding support durations x_1, x_2 and x_3. The section below provides a mathematical formulation of the problem.

Figure 2 A Visualization of Three Clashing Spacecraft Visibilities over a Ground Station



5 Model Formulation

Figure 2 displays the contiguous placement of the various activities in the general picture of a single occasion of clashing visibilities occurring at a ground station. Notice that the station must be reconfigured when support changes from one passing spacecraft to the next. The following notations are used in this representation [12]:

V = Difference of first clashing AOS and last LOS.
I = Total number of spacecraft visibilities clashing in a pass over a station ($i = 0 \ldots\ldots I$).
P = The Utility or Value Function (= $f(x_1, x_2, x_3, \ldots, x_n)$) to be maximized
a_i = Start of visibility (AOS) of spacecraft i.
b_i = End of visibility (LOS) of spacecraft i.
s_i = Start of support of spacecraft i.
e_i = End of support of spacecraft i.
x_i (= $s_i - e_i$) = Support given to spacecraft i when it passes over a station.
min = Minimum time required for support once support is commenced.
max = Maximum time required for support.
r = Station reconfiguration time reconfiguration time is added to end support of previous supported spacecraft.
C_i = Utility or profit contributed to P *per unit time* when spacecraft i is supported.

t_{i+1} is a binary variable that indicates whether spacecraft i is supported or not. Thus if $x_i = 0$, $t_{i+1} = 0$ and if $x_I > 0$, $t_{i+1} = 1$ for $i = 1, 2, 3, \ldots, I$; $t_1 = 0$.

The Optimization Model

Maximize $P = f(x_1, x_2, x_3, \ldots x_n)$

In general, the objective or evaluation function $f(x_1, x_2, x_3, \ldots x_n)$ may be nonlinear, discontinuous, and have multiple peaks (Figure 3). In the special case when $f(x_1, x_2, x_3, \ldots x_n)$ is linear,

$$P = \sum_{i=1}^{I} C_i x_i = \text{total value generated} \quad (1)$$

$$V = \sum_{i=1}^{I} x_i + (\sum_{i=1}^{I} t_i) * r = \text{total visibility at the ground station} \quad (2)$$

subject to:

(i) Start of support (s_i) of spacecraft 'i' must be at AOS_i or later and it should be equal to or less then LOS_i.

$$b_i \geq s_i \geq a_i \quad (3)$$

(ii) End of support (e_i) of spacecraft i must be at LOS_i and it should be equal to or greater than AOS_i.

$$a_i \leq e_i \leq b_i \quad (4)$$

(iii) Station Reconfiguration allowance.

$$s_i \geq \{ \underset{k=(i-1)\ldots\ldots 1}{Maximum} \ (e_k \times t_{k+1}) \} + r \quad (5)$$

(iv) Duration of support of spacecraft *i*.

$$x_i = e_i - s_i \tag{6}$$

(v) Constraint for minimum time of support. This may be either 0 or greater than the quantity *min*. Therefore,

$$x_i = 0 \ \text{or} \ x_i \geq min \tag{7}$$

(vi) Maximum time of support should be less than the quantity *max*, as specified by the decision maker.

$$x_i < max \tag{8}$$

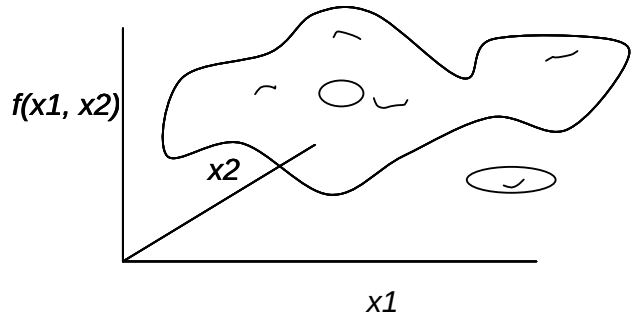
(vii) Nonnegativity constraint

$$s_i, e_i \geq 0; \quad x_i \geq 0; \tag{9}$$

Constraint (v) makes the problem nonlinear, even if the objective function (1) is linear. It may be shown that the search space is not convex. As said earlier, in general the profit or value function P (= $f(x_1, x_2, x_3, \ldots, x_I)$) may be a nonlinear and involve multiple local optima (Figure 3).

Note again that Constraint (v) implies that the feasible values of the decision variables $\{x_i\}$ are discontinuous. This is a major departure from standard LP type formulations that require convexity of the feasible space. The restrictions—constant *min*, *max* and *r*—may in general be satellite- or station-specific, when necessary.

Figure 3 An Arbitrary Profit Function of Two Decision Variables



6 An Evaluation Function for TTC Support Value Generation

In spacecraft applications it is well-known that for TTC (telemetry, tracking and command) operations the return or utility per unit support time (i.e., the marginal rate of utility generated by an additional second of support) is not uniform or constant. Rather, it depends on the *duration of support already provided* and even be discontinuous. For example, extensive discussions held with a group of space

technologists led to the following points (the values cited are only illustrative and would change to conform to the particulars of a situation).

- TTC operations require a minimum support of m minutes (m×60 seconds), whenever support is provided.
- Almost all operations can be done in this m-minute span. Thus the returns obtained in supporting a spacecraft for its *minimum support time* are high.
- A support of more then m minutes is not necessary, but if some additional visibility time is available, support may be extended beyond m minutes. The marginal utility of this "extra" support gradually diminishes with the extra support provided. Thus, providing *more than m* minutes' support is desirable, when possible, but not required.

Thus one infers that the objective function should be crafted in such a way there is no return if a spacecraft is supported for a duration < m minutes. But the return is high if support is exactly m minutes. However, the *marginal* return diminishes after m minutes. One may paraphrase this as follows. If a spacecraft is supported beyond m minutes, the returns generated are more for the "m^{th} minute" as compared to those in "m+1th minute" of support; returns in the "m+1th minute" are more than the returns generated in the "m+2th minute", etc. But the marginal return generated/minute (beyond the first m minutes) diminishes as time passes.

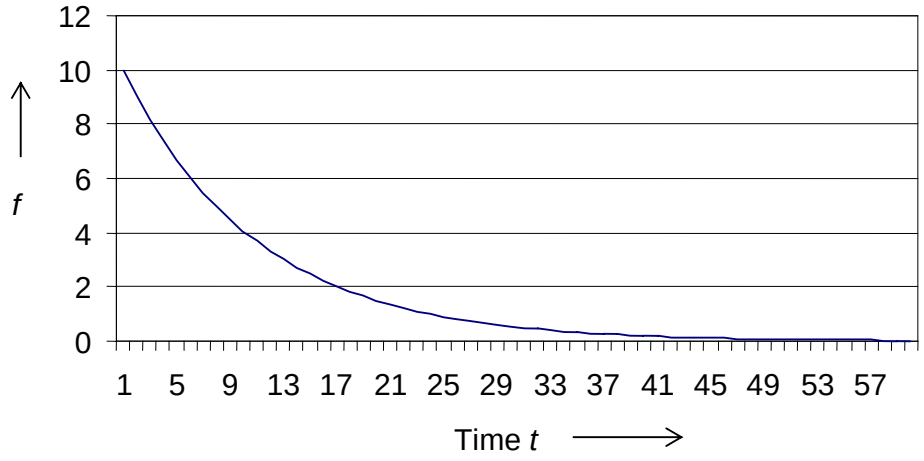
On consideration of such factors, a suitable "gradually diminishing" utility function may be designed. To aid parsimony, a one-parameter function—the *exponentially decaying* return function—is adopted for illustration. Incidentally, while this function meets all requirements listed above and is mathematically simple to manipulate, it quickly engages the space experts intuitively. Its construction would proceed as follows. A simple exponentially falling function has only one parameter λ and it may be represented as

$$f = \lambda e^{-\lambda t} \qquad (10)$$

where t is the time measured from the origin, as shown in Figure 4. At $t = 0$, $f = \lambda$, thus λ is the ordinate of the exponential curve at t = 0. To cast this diminishing function into our requirement, we introduce one more parameter, β.

$$f = \lambda e^{-\lambda \beta t} \qquad (11)$$

Figure 4 The Exponentially Decaying Marginal Return Function *f*



Here β is a factor that will control the actual decay rate of this exponential curve as support time *t* increases beyond *m* minutes.

Determining the Value of β

In this illustration we assume that the minimum support needed (*m*) is 8 minutes. Similarly, we assume that there exists an upper practical limit of providing support also (= 16 minutes), after which the marginal utility generated of additional support becomes negligible. In actual practice such input would come from the spacecraft mission experts.

Suppose that the experts indicate that the function value *f* may be assumed to reduce to 1% of its initial value (λ) when support time becomes 16 minutes. That is, when *t* = 480 seconds,

f = 1% of λ

The value of parameter β may then be found as follows.

At $t = 0$, $f = \lambda$
At $t = 480$, f = 1% of λ

Hence $\lambda e^{-\lambda * 480 * \beta} = \lambda/100$, which gives

$$\beta = 0.00958/\lambda \qquad (12)$$

Note that (11) provides the rate of *marginal* return. Therefore, at an instant *t*, the total utility generated by providing support up to *t* is the area covered between the exponential rate curve and the abscissa, easily found by integrating the exponential function [10]. Hence

Total value of support $= \int_0^t \lambda e^{-\lambda t B} dt$,

which gives

Total value of support = $(1- e^{-\lambda_t \beta})/\beta$ (13)

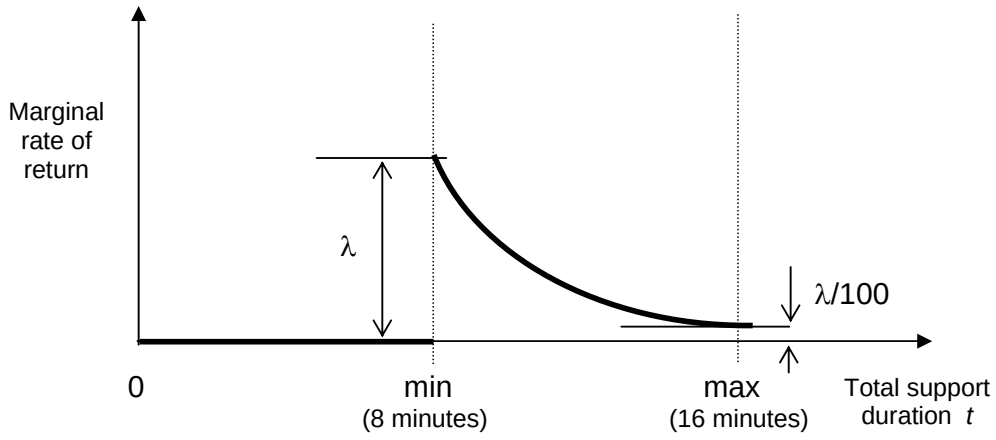
Thus, because of the minimum (8 minute) support requirement, the actual marginal rate function is not exponential, rather it is discontinuous, as shown in Figure 5.

Hence Total return = 0 for $t < 480$

= $\alpha + (1 - e^{-\lambda_t \beta})/\beta$ for $t \geq 480$ (14)

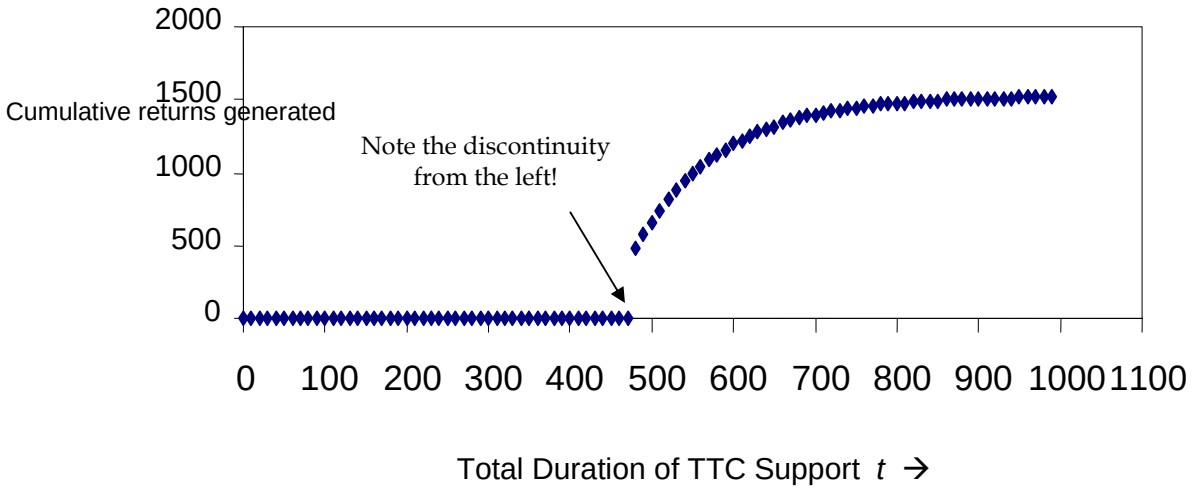
where $\alpha = 480*\lambda$, $\beta = 0.00958/\lambda$ and t = (duration of total support - 480).

Figure 5 Marginal Return vs. Duration of TTC Support



Hence one may now give the expression for the total utility of a TTC support lasting t units of time. This utility is discontinuous at the minimum support value (= m, here 480 seconds). Indeed, the term α in the expression (14) below ensures high immediate value generated when a spacecraft receives its minimum support of 480 seconds. The total return as a function of t is shown in Figure 6.

Figure 6 Total Returns as a Function of Total Support Duration *t*



7 Derivation of Lambda (λ)

The next challenge in crafting the objective function for the support optimization model is to find the value of λ. In general, λ would vary with the spacecraft, the task to be supported and the station providing the support. Numerically, λ is the rate of return when a spacecraft is supported for exactly *m* minutes (the minimum support for TTC operations). Recall that in (10) λ is the height of the original (assumed) exponentially falling utility rate curve at zero time.

The applicability of the exponential-decay logic would depend on the correct derivation of λ. In practice this would require in-depth discussion with spacecraft mission experts as their judgments would have to be accurately reflected in the quantification of λ. We present below a procedure.

Assume that the experts indicate that the value of λ for a *particular* support scenario (a combination of spacecraft, station and operation) under consideration for clash resolution would depend on the factors listed in Table 1.

Experts also indicate that these factors each have their own significance in spacecraft support and hence would each impact λ in its own manner. The experts express the relative impact of these factors as shown in Table 3—in terms of subjectively expressed priority and penalty values that reflect the various aspects of spacecraft mission objectives.

TABLE 3 SAMPLE FACTORS CONTRIBUTING TO SUPPORT CRITICALITY

FACTOR	*PRIORITY*	*PENALTY*
Maximum elevation	2	
Spacecraft priority	10	
Exclusive pass (orbit over a single station)	8	
Critically dependent operations		10
Special TTC operations (must be done in that pass)	10	
Service (visibility gap)		5
Minimum operations on the spacecraft	6	
Exactly 'n' ascending operations per spacecraft		2
Exactly 'm' descending operations per spacecraft		2
Operator constraint		2
Ground station constraint	2	

Discussions with spacecraft mission experts would typically reveal that LEO spacecraft missions have *multiple* objectives. Common among these are

- Maximization of the total number of supports provided
- Maximization of the total duration of supports
- Maximization of the execution of PL requests
- Minimization of ground station idle time
- Maximization of controller use
- Uniform distribution of tasks across stations
- Operations such as TR should be distributed globally for each spacecraft

Many other requirements could be expressed as *soft constraints* and these too should be rolled into the objective function being crafted.

Combining these considerations into one function would facilitate the valuation of λ. However, the method of combining these objectives and soft constraints should reflect their particular character and significance and their effect on mission objectives. Subsequently, factors not considered in constructing λ may be used if necessary to guide the actual selection of tasks to be supported, to enable the selection of "preferred solutions"—in the Pareto optimality sense.

For illustration we assume that discussions with mission experts have led to the deduction that λ consists of two dominant but separable parts as follows:

$$\lambda = \mathbf{Max\ (FactorExclusive, FactorCyclic, FactorPrecedent, FactorSupportGap) + FactorSpacecraft + FactorElevation} \quad (15)$$

FactorExclusive, FactorCyclic, and FactorPrecedent are expressions of the character of the task to be performed. FactorSpacecraft is also an expression of importance or priority. In practice these factors may each be subjectively quantified in consultation with mission specialists as weights, each set on a scale of 1 to 10. The other factors in (15) depend on *dynamically developing* conditions. For instance, FactorSupportGap would depend on the time gap between current visibility and previous supported visibility for the same spacecraft. FactorElevation is step function assumed for illustration to be rising from a value of 0 (for the maximum elevation of the visible pass $\leq$ say 5°) to 10 (for elevations $\geq$ 5°).

Note again that the above expression for λ has *two* parts. The first part is the *maximum* of four factors while the second part is *added* to it. It is conspicuous that the first part has terms that correspond to the actual *operation* performed on the spacecraft. By contrast, FactorSpacecraft and FactorElevation relate to (a) the *spacecraft* concerned and (b) the *quality* of the visibility window and are both *independent* of the actual operations.

Furthermore, the first part of λ has factors that are each apparently very important in determining support/no support decisions. Hence, if any one factor among these is high for a visibility, that visibility becomes a high contender and it should be supported. This requirement is ensured by the construct of λ as shown by (15), for it takes the *maximum* of Factor Exclusive, FactorCyclic, FactorPrecedent and FactorSupportGap.

We provide below illustrative valuations of the factors in λ.

Subjective Setting of the Factors in λ

FactorExclusive: This factor assumes only two values, either 0 or 10. If a visibility is exclusive in that specific orbit (i.e., this spacecraft is not going to be visible for at least for the full duration of the next orbit), this factor becomes 10. Otherwise it is 0. Thus, if a visibility is exclusive, this factor increases the value of λ in order that the likelihood of support for this visibility rises.

FactorCyclic: This factor also takes values of 0 or 10. If a cyclic operation falls in this slot of visibility, FactorCyclic becomes 10. Otherwise it is 0. Thus, if a spacecraft requires a cyclic operation to be supported in this particular visibility, this factor becomes 10 and thus it increases the value of λ and hence the likelihood of providing support for this visibility becomes high.

FactorPrecedent: This factor also takes values of 0 or 10. If a *precedent* operation (required by a subsequent *dependent* operation) falls in this slot of visibility, this factor becomes 10, otherwise it is 0. Thus by giving precedent operation a value of 10 we raise the odds that the dependent operation will have its precondition met.

Such "chaining" effect develops due to the fact that for certain spacecraft a precedent operation's completion is the prerequisite for the dependent operation.

FactorSupportGap: Generally speaking, a spacecraft needs to be health-monitored, telemetered, etc. throughout the day. Consequently, an important requirement in spacecraft support scheduling is to keep the *gap between supports* small. FactorSupportGap ensures that a spacecraft will have an increased likelihood of support after a certain time gap. The value of this factor is dynamically set and it is kept dependent on the time gap between current visibility and previous supported visibility for the same spacecraft. As this gap increases (because a support has not yet been possible), the value of FactorSupportGap also builds up. After some delay (i.e., gaps piling up) this factor will build to such a high value that it will exceed any other priority factor in its numerical value. Thus the affected spacecraft will end up receiving a support before too long.

FactorSpacecraft: This factor is a spacecraft-dependent factor and is independent of operations and visibility. This factor is an additive term in λ. This factor indicates that when a conflict develops, the "more important" spacecraft (as designated by mission management) should have a higher probability of getting a support. An example is a new spacecraft or a launch that needs to be supported at every possible opportunity. When FactorSpacecraft for this spacecraft is set high, it will preempt the support of any other normal spacecraft.

FactorElevation: This factor is defined as a step function. In general, this factor is also independent of operations, hence it is additive in λ. This factor indicates that a higher elevation visibility is more suitable for doing an operation, regardless of the operation, as compared with a low elevation visibility.

We must point out that the above factors would now *control* the priority with which a particular operation on some particular spacecraft will be executed when a clashing scenario develops. Hence, the values assigned to these factors must reflect the best judgment of mission specialists and mission objectives.

We remark again that due to the manner in which we developed the foregoing logic of determining the priority of support, these factors alone would control the value of λ and hence the quality of the overall support schedule. Note, however, that this is only illustrative of a general procedure that may be similarly set up.

8 The Objective Function for Payload Operations Support

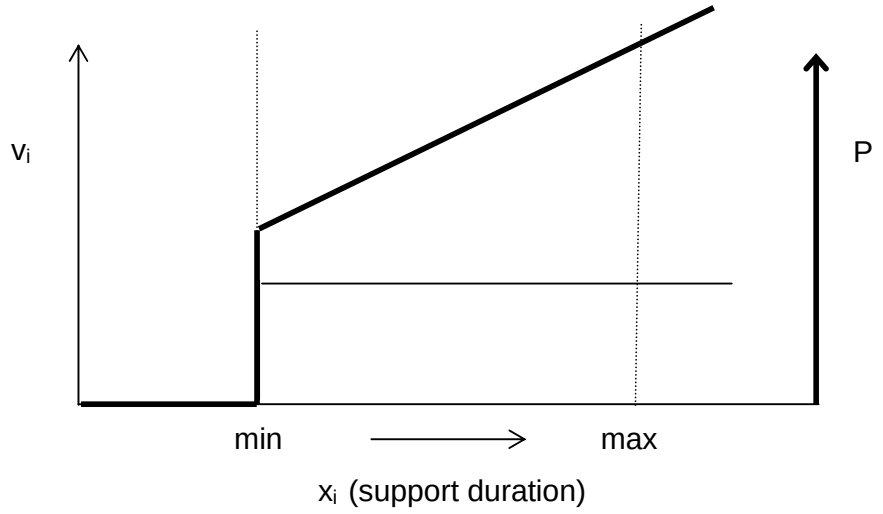
For Payload operations, the marginal return of additional support (or the return per unit time) is generally constant, regardless of how much support has already been *provided. Hence, the evaluation function becomes a linear objective function for the PL clash resolution problem. It is then simply the product of the support time (x_i) and returns per unit time (v_i), beyond any minimum support required. The

parameter v_i is dependent on the spacecraft and the kind of PL operation being performed. Thus the total utility from supporting several spacecraft in a visibility is

$$P = \sum_{\text{over all i}} (v_i \times x_i) \quad (16)$$

The function is shown in Figure 7.

Figure 7 Marginal (v_i) and Total Utility (P) for PL Support



9 Optimization of TTC Visibility Clash Support

If there are *I* spacecraft passing over a given ground station simultaneously, solving the following problem would produce the optimal allocation of support to each spacecraft:

Maximize,

$$P = \sum_{\text{over all i}} \{\alpha_i + ((1 - e^{-\lambda_{it}\beta_i})/\beta_i)\} \quad (17)$$

Subject to

i) Start of support of spacecraft '*i*' must be at AOS_i or later and it should be *equal to or less* then LOS_i.

$$b_i \geq s_i \geq a_i \quad (18)$$

ii) End of support of spacecraft '*i*' must be at or before LOS_i and it should be *equal to or more than* AOS_i.

$$a_i \leq e_i \leq b_i \quad (19)$$

iii) Reconfiguration allowance.

$$s_i \geq \{ \underset{k=(i-1).....1}{Maximum} \ (e_k \times t_k) \} + r \qquad (20)$$

iv) Duration of support of spacecraft *i*.

$$x_i = e_i - s_i \qquad (21)$$

v) Constraint for minimum time of support. Support provided may be either 0 or greater than *min*:

$$x_i = 0 \ \ or \ \ x_i \geq min \qquad (22)$$

vi) Constraint for maximum time of support. Support should be equal to or less than *max*.

$$x_i \leq max \qquad (23)$$

vii) Non negativity constraint

$$s_i, e_i \geq 0 ; \quad x_i \geq 0 ; \qquad (24)$$

The objective function (17) and the presence of constraint (22) make the problem non-linear.

In the above formulation, for convenience and without loss of generality we made the assumption that reconfiguration times are constant and have same value 'r' for all sets of ground stations and spacecraft.

Objective functions similar to (16) and (17) have already been implemented to resolve visibility clashes optimally in a prototype spacecraft support scheduling system. (16) models the utility objective for PL visibility support while (17) models the objective for a TTC support.

Solving the above problem would be nontrivial in general for it is NP-complete. One approach to produce a practical solution in reasonable amount of time would be to use a heuristic or device that would reduce the effort to be expended in seeking a good solution. For other similar problems meta-heuristic search methods—tabu search, simulated annealing or genetic algorithms—have been shown to be quite effective [14]. We approach the problem as follows.

10 An AI-based Formulation of the Optimization Problem

Before we present meta-heuristic or AI-based methods for solving the above problem, a brief recollection of the utility of such methods would be in order. Methods such as greedy search [16] perform local search as opposed to a global search, which is what is required when the object is to solve an optimization problem globally. Local search attempts to find a solution better than the current one through a search only in the neighborhood of the current solution. Two solutions are neighbors if one can be obtained through a well-defined modification of the other. The greedy search method can be programmed using conventional techniques to look for a neighbor that does least or no damage to feasibility and most helps the objective function. However, working alone and given a starting point, such a procedure cannot guarantee reaching the globally optimal solution. It often gets trapped in a local optima with no way available to it to escape from it.

A meta-heuristic is a top-level general strategy which guides other heuristics to search for feasible solutions in domains where the task is hard. Meta-heuristics have been most generally applied to problems classified as NP-hard or NP-Complete. However, meta-heuristics would also be applied to other optimization problems for which it is known that a polynomial-time solution exists but is not practical. If the neighborhood generation method is intelligent, repeated a reasonable number of times a good local search procedure combined with appropriate randomization can reach a solution of acceptable quality, even if the final solution not globally optimal. Indeed this is how many complex practical optimization problems are now being tackled, particularly in scheduling [14] and in engineering design optimization [4]. Three of the most popular methods for improving search are tabu search, simulated annealing and genetic algorithms. Such methods belong to the domain of artificial intelligence, for they incorporate nontrivial moves, learning, and the capability to automatically modify the direction of search based on emergent information. What separates these "AI" methods from conventional computer programs is that these methods are not programmed to execute instructions along pre-destined paths.

The key idea in each of these meta-heuristic search methods is to innovate procedures that minimize nonimproving moves to neighborhood. They do this by making provisions to prevent repeating solutions. Tabu search deals with cycling by temporarily forbidding moves that would return to a solution recently visited. The effect prevents short-term cycling, although solutions can repeat over a longer period. Simulated annealing controls cycling by accepting even nonimproving moves according to probabilities tested with computer-generated random numbers. A parameter "temperature" controls the randomness of the search. Genetic algorithms (GAs)—another still very popular metaheuristic to apply to complex optimization problems—evolve good heuristic solutions by performing "genetic operations" on a population of solutions that when properly parameterized continually improves as the generations go by; GA breeds new solutions by combining existing solutions. GA search is more general and abstract than both simulated annealing and tabu search [14]. Newer GAs are multi-objective [4].

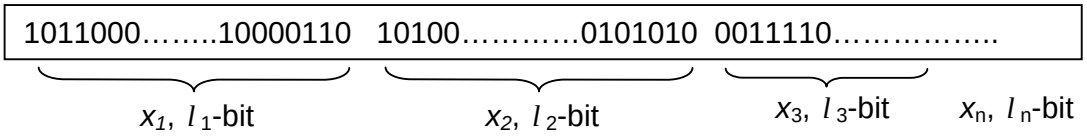
Such methods have the advantage that they can be applied to a problem without requiring much knowledge about the structure of the problem [14]. Further, these methods, particularly GA, may often be hybridized with other heuristic "solution improvement" methods that exploit domain-specific knowledge. (This is now frequently done in scheduling [4].) However, the computation time needed to obtain a solution by a meta-heuristic method tends to be relatively long and therefore, such methods remain presently the last resort when no exact method is available.

In the clash resolution problem formulated in this paper, the decision variables are $\{x_i\}$. However, due to non-linearity and non-convexity present, the general problem cannot be solved by traditional optimization methods. As already noted, the general problem cannot be modeled as an LP due to its non-linearity. We outline a solution by genetic algorithms, a technique now widely described in the literature.

The GA Chromosome Construction

A key step in using GAs is casting the solution to the optimization problem in a *chromosome-like* structure in order that "genetic operations"—crossover, mutation, selection, etc.—may then be performed on it. We often use binary (0-1) coding to construct this chromosome because in many problems the decision variables $\{x_i\}$ are real numerical quantities while binary coding is easily manipulated by straightforward genetic operators. A single chromosome for spacecraft support would contain all information for the full duration of support for each spacecraft involved in a clashed set of visibilities at a station.

An example of such coding would be as follows. Suppose that at some ground station four spacecraft visibilities are clashing. Then, the chromosome would contain four decision variables—x_1, x_2, x_3, and x_4. In general, for n visibilities clashing and in need of value-maximized clash resolution, the chromosome itself would have the following appearance (note the difference in the bit lengths $\{l_i\}$ of the different decision variables):



In GA formulations using binary-coded variables, the length of a substring representing a real variable (such as x_i) depends on the numerical accuracy desired of the final solution. If we us ℓ_i binary (0-1) bits to code the variable x_i the obtainable accuracy in that variable would be approximately [7] $\left(x_i^{(U)} - x_i^{(L)}\right)/2^{\ell_i}$. In most remote sensing LEO spacecraft operations the visibilities are about 20 minutes

or less, within a second. So, to illustrate a case, if 10 bits are used to represent x_i}, the obtainable accuracy would be of the order of

$$(16*60+23)/2^{10} = 0.959$$

seconds. Thus, when sufficient time is given to the GA converge, the implementation of this level of precision (i.e., ℓ_i = 10 bits) in chromosome coding would keep the numerical error of the final solution limited to one second or less.

Feasibility Assurance and GA Operators

In nature, the feasibility of a species is maintained by nature's incorporating constraints inside chromosome coding. (A violation of these constraints by say mutation may cause cancer-like cell formation.) In the spacecraft support problem feasibility of solutions may be ensured by imposing suitably modified upper and lower limits of possible values of support times $\{x_i\}$ [12]. Figure 2 indicates such limits.

Formally, this may be stated as follows. Consider the object of ensuring feasibility of two decision variables x_1 and x_2 (refer to Figure 2). The limits for x_1 are $[a_1 , b_1]$. Limits for x_2 will be [(Greater of e_1+r and a_2), b_2]. Such approach would ensure feasibility of all chromosomes. Experience shows that putting such constraints *in the chromosome structure itself* reduces the effort in optimization. Otherwise numerous penalty functions would have to be incorporated in the objective function.

GA facilitates global search by creatively generating new solutions as the search progresses. In GA, "crossover" creates progeny by exchanging information (gene-holding *segments* of the chromosome) among selected parent strings resident in the mating pool [7]. "Mutation" facilitates local search. Various "selection" strategies influence the selection of "parent" chromosomes that subsequently procreate the next generation. The preserve-the-best and roulette wheel selection strategies [7, 4] were implemented in the numerical example given below.

11 A Numerical Example with Linear Objective function

This example assumes that payload (PL) support is to be provided to four spacecraft simultaneously passing over a ground station supporting them, causing their visibilities to clash. The maximum support required is uniformly 983 seconds while the minimum support required, once support is decided to be given, is 263 seconds. The visible time windows are as follows (all expressed in seconds from a reference point):

a_1 = 616520, b_1 = 617115; a_2 = 617244, b_2 = 618318;
a_3 = 617712, b_3 = 618650; a_4 = 618027, b_4 = 619196

The above data (616520, 617115, etc.) are time marks (in seconds) from a reference point. The profit function $f(x_1, x_2, x_3, \ldots, x_I)$—here a linear function involving marginal value generation rates C_1, C_2, C_3 and C_4 (the respective profits generated per unit time of PL support)—is $\sum_{i=1}^{I} C_i x_i$.

To solve this problem a bit length of 10 was used for each of the four decision variables x_1, x_2, x_3 and x_4. The GA parameters p_c (the probability of crossover) and p_m (the probability of mutation) were set at 1.0 and 0.01 respectively, values determined as optimal by conducting pilot GA runs in a design-of-experiments framework [20]. Population size was held at 20. The GA converged to within values of 0.4 seconds when run for 100 generations, a precision sufficient for the application at hand. Table 4 displays the support times determined for four different problem scenarios, each with its own $\{C_i\}$ profile. A reconfiguration time (r) of 600 seconds was assumed to be in effect whenever the station support switched from one spacecraft to another.

Intuitively, given the unequal value generation rates (C_i) of the different spacecraft and a linear profit function, we would expect the spacecraft with highest C_i to receive maximum support. This has a caveat, however. Because a non-value generating task of reconfiguration is involved whenever support is switched from one spacecraft to another, it is probable that it would be optimal to *not* support some particular spacecraft at all.

Table 4 Optimal Support Schemes for Maximizing Profits

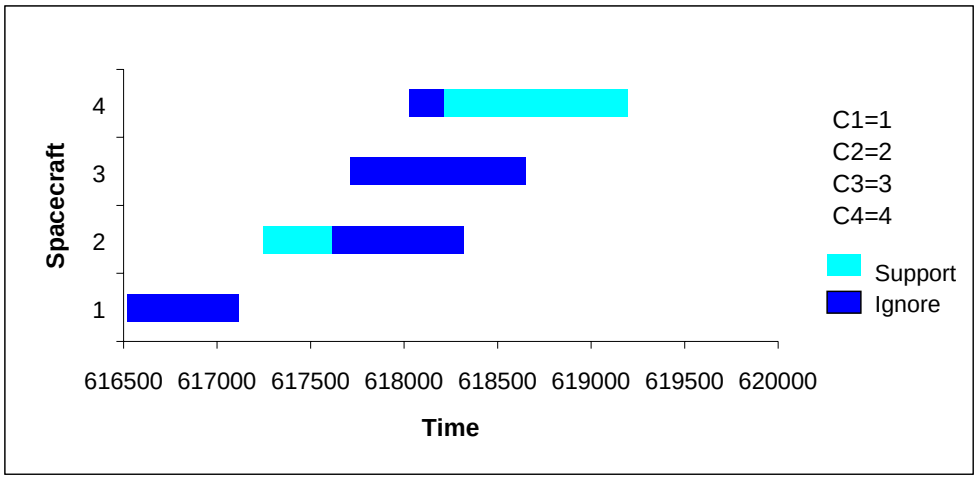
Scenario 1	s_i	e_i	*Support* (x_i)	*Total Profit*
$C_1 = 1$	616520	616520	0	
$C_2 = 2$	617244	617613	369	
$C_3 = 3$	618213	618213	0	
$C_4 = 4$	618213	619196	983	4670
Scenario 2				
$C_1 = 4$	616520	617115	595	
$C_2 = 3$	617715	618318	603	
$C_3 = 2$	617712	617712	0	
$C_4 = 1$	618918	619196	278	4467
Scenario 3				
$C_1 = 1$	616520	616520	0	
$C_2 = 4$	617244	618227	983	
$C_3 = 3$	617712	617712	0	
$C_4 = 2$	618827	619196	369	4670
Scenario 4				
$C_1 = 1$	616520	617112	592	
$C_2 = 3$	617712	617712	0	
$C_3 = 4$	617712	618650	938	
$C_4 = 2$	618027	618027	0	4344

Each scenario in Table 2 was solved by the GA in about 3 seconds by a C++ program running on a Pentium III 500 mHz system. Observe that support does shift towards the spacecraft contributing at the maximum rate to total profits. Also, some spacecraft get no support at all. The estimated total profits are also shown in each case. Figure 8 graphically displays the spacecraft visibility and support periods for the profit rates $\{C_i\}$ of Scenario 1. In a modern network operating a dozen LEOs and 20-odd ground stations, up to 200 or more such PL visibility clashes would need to be optimally resolved every week before the operating schedules are released to the different ground stations spread around the globe. GA would take about 3 minutes on a Pentium III system to resolve the 200-odd clashes.

No exact method is known to exist in the literature to analytically solve a resource allocation problem of such complexity.

Is GA the best method for solving this problem? There are several criteria on the basis of which this may be judged. GA, simulated annealing and tabu search, each produce comparable final solutions. GA takes the longest time and tabu search the shortest. However, there is one criterion on the basis of which GA may get a preferential nod. GA produces a *family* of final solutions whereas both simulated annealing and tabu search produce single solutions. There may be other, perhaps qualitative, criteria also to judge the acceptability of the final support schedules. Such *multiobjective* situations may also be addressed by the GA methodology with suitable modification of the chromosome structure and an extension of GA's selection logic [4]. This can produce Pareto optimal final schedules.

Figure 8 Satellite Support Periods for Profit Scenario 1



12 Support Optimization with Exponential Objective function

This section illustrates the method for optimal clash resolution when the objective function is nonlinear, in particular, when it has an exponentially decaying marginal return character as derived in Section 6. Suppose that TTC support is to be provided to two spacecraft simultaneously passing over a ground station supporting them, causing their visibilities to clash. The maximum support required is uniformly 1800 seconds while the minimum support required, once support is decided to be given, is 480 seconds. The visible time windows are as follows (all expressed in seconds from a reference point):

$a_1 = 0, \quad b_1 = 840; \quad a_2 = 900, \quad b_2 = 1800$

Station reconfiguration time is 10 minutes or 600 seconds. The profit function for each spacecraft is $f(x_i)$—here an exponential function involving parameters α_i, β_i and λ_i (see Section 6)—given by

$$\text{Profit} = 0 \qquad \text{for } t < 480$$

$$= \alpha_i + (1 - e^{-\lambda_i t^{\beta_i}})/\ \beta_i \qquad \text{for } t \geq 480$$

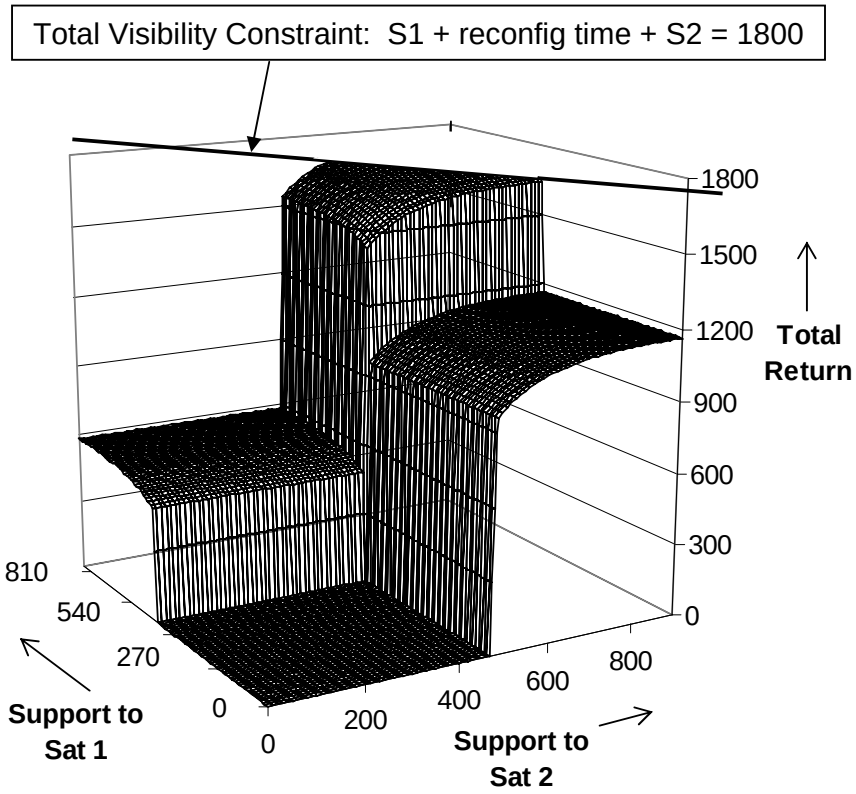
where $\alpha_i = 480*\lambda_i$, $\beta_i = 0.00958/\lambda_i$ and t = (Duration of total support - 480).

GA was parameterized as population size = 20, probability of crossover = 0.95 and probability of mutation = 0.1, and the number of generations to run = 100. We also point out that GA uses randomization in several places, including mutation, crossover, and also selection. The random path that a particular GA execution follows is controlled by the seed of the random number that initiates GA. A robust GA would produce good results regardless of the value of this initial seed. For the example solved, five GA replications produced the following results:

Random Seed	**1**	**2**	**3**	**4**	**5**
Spacecraft 1 Start of support	0	0	0	0	0
Spacecraft 1 End of support	601.013	600.124	601.698	602.176	604.882
Spacecraft 2 Start of support	1201.01	1200.12	1201.69	1202.18	1204.88
Spacecraft 2 End of support	1799.91	1799.98	1799.97	1799.99	1799.99
Total return generated	1102.61	1102.63	1102.62	1102.62	1102.56

The data indicate both good convergence as well as robust performance of the GA. A question may be asked, how difficult is the return maximization task here? We address this question for a more complicated scenario when the two spacecraft-task combinations have different λ.

Figure 9 Total Return as function of Support Times S1 and S2



Let $\lambda_1 = 1.0$ and $\lambda_2 = 2.0$. Figure 9 displays the behavior of the (total) objective function (the total return is produced here by partially supporting spacecraft 1 and spacecraft 2). We generated the data here by enumeration. The solution space is restrained by the constraints

(a) Total visibility equals 1800 seconds and
(b) Station reconfiguration time is 600 seconds.

Observe two aspects of the function plotted in Figure 9: (1) Strong nonlinearity exists in the objective function, and (2) discontinuities exist where the objective function sharply jumps to a different value. An application of GA produced a solution that maximizes total return as follows:

Spacecraft 1 Start of support 0
Spacecraft 1 Endof support 564.71
→ S1 = Support of spacecraft 1 = 564.71 seconds

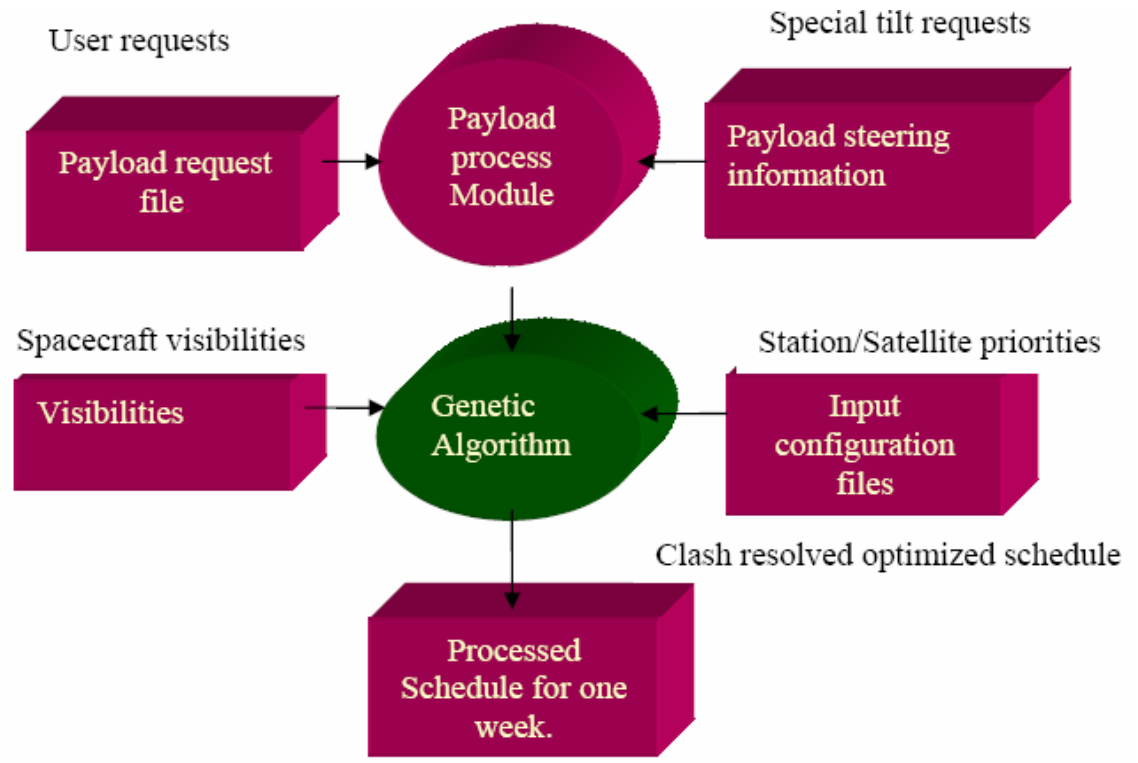
Spacecraft 2 Start of support 1164.71
Spacecraft 2 End of support 1799.96
→ S2 = Support of spacecraft 2 = 635.25 seconds

We verify that the total support time + the reconfiguration time is 1799.96 seconds, a sufficiently good practical solution. Also, it is not difficult to see that the GA-generated solution falls in the region where total return is highest *subject to the constraints* (a) and (b).

13 An Application

This section provides the summary of an implementation based on the visibility clash resolution logic developed above and the consequence (benefits) observed. In this implementation, customer requests for payload data are processed for special tilt requirement, sensor selection etc and feasibility of scheduling those passes are first studied. These processed requirements, and the visibilities of all the spacecrafts, over all the stations, serve as input for optimization. The configuration files (containing the scheduling environment and resource characteristics) serve as the controlling factor for operation allocation. The Genetic Algorithm is applied at this level to optimally allocate the operations for a duration of one week, by resolving all visibility clashes and controller clashes occurring. All these processes (modules within the "Scheduling System", Figure 10) are automated and are to be utilized for schedule generation on a weekly basis.

Figure 10 Functional Modules of an implemented Satellite Support Scheduling System



A sample output of support schedules generated is given below. The clashing passes over Mauritius ground station (MAU) are highlighted for easy reference. In this, the visibilities of spacecraft IRS-1D, IRS-1C, IRS-P4 and IRS-T1 are clashing. The scheduling system has optimized the allocation of the passes and supports IRS-1C and IRS-T1 only. The decision took less than 100 generations of the GA executed in a fraction of one second on a Pentium IV. Note that all these calculations are performed off-line and the support decisions are programmed into the electronic infrastructure well ahead of the execution of the actual support. Note also that the GA parameters were optimized before "production" execution using statistically designed factorial experiments for a variety of clash topologies and payoff scenarios (Figures 1, 5 and 9).

Table 5: Sample Optimized Spacecraft Support Schedule after Resolution of Clashes

Date	*Space-craft*	*Stn*	*Orbit #*	*Max elevation*	*AOS hh:mm:ss*	*LOS hh:mm:ss-*	*Operations Required*
2004 03 14	I1D	SN1	33773	19.18	06:10:43	06:16:20	PL, P85 (TILT= -2.07), G3
2004 03 14	I1D	BLE	33773	17.99	06:11:56	06:24:40	TM, TC, TR
2004 03 14	I1C	LK1	42612	7.29	06:13:19	06:22:26	TM, TC$
2004 03 14	I1C	SN2	42612	5.31	06:17:20	06:24:10	NO Support
2004 03 14	I1C	BLW	42612	4.38	06:18:19	06:26:29	TM, TC
2004 03 14	I1D	**MAU**	33773	70.78	**06:21:14**	**06:35:51**	***NO Support***
2004 03 14	IT1	DLH	13111	31.35	06:24:45	06:36:03	NO Support
2004 03 14	IP4	LK1	25434	52.26	06:27:26	06:37:55	TM, TC#
2004 03 14	IT1	LK2	13111	19.94	06:25:20	06:35:59	TM, TC, TR, PLDW, DSS
2004 03 14	I1C	**MAU**	42612	24.90	**06:26:07**	**06:35:38**	TM, TC, TR$
2004 03 14	IP4	BLE	25434	48.46	06:27:47	06:41:46	TM, TC, TR, PB, PYS
2004 03 14	IP4	SN1	25434	47.77	06:28:00	06:37:00	OCM_RT, P10 (TILT= -20.0)
2004 03 14	IT1	SN1	13111	19.29	06:28:03	06:38:39	NO Support
2004 03 14	IT1	BLW	13111	18.38	06:29:29	06:39:41	TM, TC#$
2004 03 14	IP4	**MAU**	25434	12.84	**06:39:22**	**06:50:42**	***NO Support***
2004 03 14	IT1	**MAU**	13111	66.89	**06:40:38**	**06:50:09**	TM, TC, TR, PB#
2004 03 14	IP6	BR1	2119	15.82	06:54:08	07:07:12	TM, TC, PB
2004 03 14	IP6	LK2	2119	7.43	07:01:28	07:11:29	TM, TC
2004 03 14	IP6	SN3	2119	5.41	07:04:40	07:11:52	PL, P76, T02.27, RT, L4L3AW
2004 03 14	IP6	BLW	2119	4.46	07:06:28	07:14:42	TM, TC

A new scheduling system coded into software in VC++ based on the logic presented in the foregoing sections is currently under evaluation at an actual applications site. The results are being cross-compared with schedules produced by an experienced team of scientists well familiar with multi-satellite schedules generation on a regular basis. A sample of the results of this comparative study appears in Table 6.

Table 6 Comparison of Manual Method vs. GA-bases Optimization—How many extra passes got scheduled in one week

Spacecraft	*No. of Passes Scheduled by Manual Scheduling using local intelligence (heuristics)*	*No. of Passes scheduled by GA Method*	*Increase in the number of passes supported*
IRS-1C	126	136	10
IRS-1D	125	132	7
IRS-P4	119	122	3
IRS-T1	107	106	-1
IRS-P3	144	149	5
IRS-1B	38	36	-2
IRS-P2	22	27	5
IRS-P6	126	139	13
Total	807	847	40

The last column in Table 6 indicates the impact of using GA to resolve clashes. On close examination it was revealed that the manual method of scheduling has not correctly exploited the total value generated by a support. It is clear, nevertheless, that nearly 5% extra passes could be supported by the new method for the considered fleet of eight spacecraft contending simultaneously for the available ground support resources. The opportunity for increasing such "performance" is dependent on the number of passes that involve visibility or chain clashes. As the number of orbiting spacecraft increases, such opportunities go up. For instance, if a fleet has only 2 or 3 satellites, clashes may be rare, unless the spacecraft are close neighbors. On the other hand, if the fleet consists of 10 or fifteen spacecraft, clashes, and therefore the opportunity to optimize the support allocation, would also be high. Such tests are currently in progress. The additional benefits of automating clash resolution that was demonstrated was (a) using only 250 CPU seconds on a Pentium IV to complete all clashing and nonclashing passes of a fleet of about a dozen spacecraft, (b) inclusion of two or more chains at a ground station in sensitivity studies, (c) be as realistic as possible in formulating the payoff functions to achieve near-optimal tradeoffs, and (d) test the feasibility of using limited manpower to accomplish tasks that earlier required laborious data manipulative effort.

14 Concluding Remarks

The central challenge in optimizing spacecraft support appears to be the handling of discontinuities and nonlinear objective functions and constraints. This paper reports on a successful scheduling scheme developed to increase the utilization of capital-intensive space equipment, and to improve customer service. Spacecraft now constitute a key underpinning for natural resource management, communication, flood control, crop output estimation, defense, and other uses worldwide. This paper has presented methods and established their real life utility for optimally resolving visibility clashes—situations met frequently in scheduling payload (PL) ground support to LEO spacecraft. Since PL activity generates real revenue from

LEO networks, its optimization has critical value for the enterprise. For TTC (Teletype, Telemetry and Communication) also, this problem occurs several hundred times/week in a typical LEO network and hence should be optimally resolved.

Such resource contention scenarios exist also in motion picture exhibition and in scheduling advertising in printed and Internet media. We recall that for certain special conditions the problem may be solved exactly. For the more general situation involving arbitrary objective functions and nonlinearity in constraints, no exact solution methods are known to exist. This paper devises and then demonstrates the efficacy of an AI-based search method, here GA, to solve the problem.

Likewise, once we have a way to determine optimal support given a set of resources, the fraction of visibilities that go unsupported after the optimal allocation of support may also be found. It would then be entirely possible for planners to introduce *virtual capacities* to find how much additional station or antenna capability (or even *new* ground stations) can be justified to support the spacecraft constellation. This would be an additional use of good, practical solution approaches such as the method given here. Till now such issues could not be adequately addressed [15].

While we have demonstrated the utility of AI-based search methods in spacecraft support scheduling, AI cannot yet match the human mind. As of this writing, machines can understand language, learn, reason, solve problems—all given *formal* rules. But machines do not imagine new scenarios nor can they automatically create *new* solution methodologies. Such capabilities must await our acquiring deeper insights into the mind's functioning. We ran into such issues in this study when we were attempting to pin down "What are we trying to optimize? What is the objective function?" (Section 7 above). Machines do not respond to such questions, though they can now do limited inductive reasoning; they can "learn" and "adapt" by automatically modifying certain logic control parameters and constructs. The goal of AI is Turing's Dream: to build a digital mechanism that would accomplish some task that the public thinks requires qualities characteristic of the human mind: plasticity, intelligence, flexibility, communicability, etc. [26]. This counters the use of small specific methods to assist in database searches, airline reservations, graphic display handling, and the like, and also to do tasks that require brute computational force surpassing human capacity even if few are impressed anymore by hand-held devices that do calculations at lightning speed. Chess-playing programs, for instance, are becoming increasingly more powerful simply by increasing the amount of brute force applied.

Methods such as those applied in this paper have one objective: They provide intelligence in the use of such brute force—to help solve problems that otherwise we would sidestep. For such problems, without at least a usable meta-heuristic at hand, a solution may be of unknown quality at best and pretty bad at worst.

14. Acknowledgements

The authors are grateful to P Soma of the Indian Space Research Organization (ISRO) and his staff who shared the problem with the author and provided critical feedback throughout the duration of this study. The authors also appreciate the efforts of Garima Shahi and Sagar Kapse—research scholars at the Indian Institute of Technlogy Kanpur—who coded, tested and finally developed the prototype scheduling system that was used by ISRO to compare its performance to manually developed spacecraft support schedules. ISRO put this system on line in production mode in 2004.

15. References

[1] Agnese, J C, Bataille, N, Blumstein, D, Bensana, E and Verfaillie, G (1995). Exact and Approximate Methods for the Daily Management of an Earth Observation Satellite, *Proc 5th Workhsop ESTEC Artificial Intelligence and Knowledge Based Systems for Space*, Noordwijk (NL), 10-12 October.

[2] Agnese, J C and Brousse, Pascal (1998). Scheduling Techniques for a Constellation Visibilities, Spaceflight Dynamics; *Proc AAS/GSFC International Symposium*, Greenbelt, MD, May 11-15.

[3] Bagchi, T P and Deb, Kalyanmoy (1996). Calibration of GA Parameters: The Design of Experiments Approach, *Computer Science and Informatics*, 26, 3.

[4] Bagchi, T P (1999). *Multiobjective Scheduling by Genetic Algorithms*, Kluwer Academic.

[5] Baker, K R (1993). *Elements of Sequencing and Scheduling*, Dartmouth College.

[6] Eliashberg, J, Swami, Sanjeev, Weinberg, C B and Wierenga, B (2000). Implementation and Evaluation of SilverScreener: A Marketing Management Support System for Movie Exhibitors, Working paper, U. Pennsylvania.

[7] Goldberg, D E (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley.

[8] GREAS, The Satellite Resource Management Tool, www.psrw.com/GREAS/grweb2.html

[9] Groleau, N, Kiser, L and Girouard, F (1992). A fully implemented semi-automated ground-control system for the Terriers satellite, NASA Ames Research Center.

[10] Kapse, Sagar Ramesh (2001). *LEO Satellite Visibility Clash Resolution using Greedy Search, Tabu Search, Simulated Annealing and Genetic Algorithms*, M Tech Thesis, Indian Institute of Technology Kanpur.

[11] Kasturirangan, K (2000). Development in Indian Space Programme, *Technorama* (IE India), August, pp. 5-9.

[12] Kumar, Sanjay (2000). *LEO Satellite Operations Scheduling: An Approach using Genetic Algorithms*, M Tech Thesis, Indian Institute of Technology, Kanpur, 2000.

[13] Pemberton, J C and Galiber, III, F (1998). A constraint-based Approach to Satellite Scheduling, Pacific Searra Research.

[14] Pinedo, M (1995). *Scheduling: Theory, Algorithms, and Systems*, Prentice Hall, NJ.

[15] Rao, J D, Soma, P and Padmashree, G S (1998). Multi-Satellite Scheduling System for LEO Satellite Operations, Paper 2b002, *Proceedings*, Space Ops 98.

[16] Rardin, Ronald L (1998). *Optimization in Operations Research*, Prentice-Hall.

[17] Shahi, Garima (2001). *A Constraint Satisfaction Heuristic for Scheduling Telemetry, Tracking and Commanding Operations of Indian Remote Sensing Satellites*, M Tech Thesis, Indian Institute of Technology Kanpur.

[18] Sousa J P and Wolsey, L A (1992). A Time Indexed Formulation of Non-preemptive Single Machine Scheduling Problems, *Mathematical Programming*, 54.

[19] SPIKE: AI Scheduling for Hubble Space Telescope, www.stsci.edu/apsb/doc

[20] Swami, S, Eliashberg J and Weinberg, C B (1999). "SilverScreener: A Modeling Approach to Movie Screens Management," *Marketing Science* (Special Issue on Managerial Decision Making), 18 (3), 352-372.

[21] Swami, S (1998). *Dynamic Marketing Decisions in the Presence of Perishable Demand*, Ph D Thesis, U. Brititish Columbia.

[22] Williams, D N (1997). *Time Indexed Formulation of Scheduling Problems*, M Sc Thesis, U. British Columbia.

[23] Wolfe, W J and Sorensen, S E (2000). Three Scheduling Algorithms Applied to the Earth Observing Systems Domain, *Management Science*, January.

[24] www.monet.astro.uiuc.edu/adass98/proceedings/kleineresc/

[25] www.vs.afrl.af.mil/fy99sbir/html/vss9cw02.html

[26] http://www.stanford.edu/group/SHR/4-2/text/introduction.html